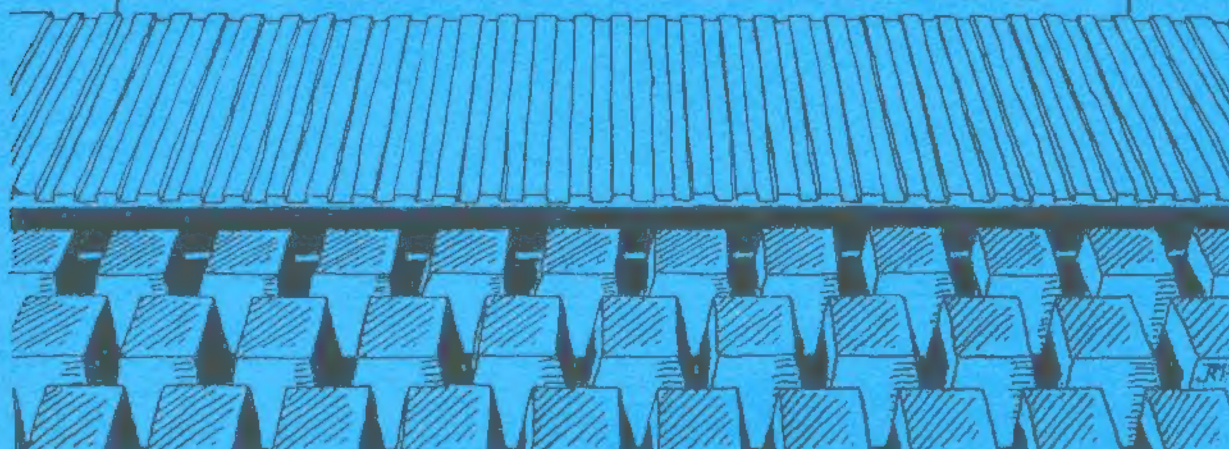
ACORN NIEUWS

JAARGANG
nummer

```
 10 PROGRAM SCRAMBLE
 20 DIM S30,AA(30,2)
 30 AA(0,0)=31;READ $S;H=(32-LEN(S))/2
 40 CLEAR 0;MOVE (H*2),19;PLOT 1,(LEN(S)*2-1),0
 50 FOR I=1 TO LEN(S)
 60   DO AA(I,0)=ABSRND%LEN(S);AA(I,1)=ABSRND%LEN(S)
 70     AA(I,2)=SGN(AA(I,0)-AA(I,1))
 80   UNTIL AA(I,2)<>0 AND AA(I,1)<>AA(I-1,0)
 90   FOR J=AA(I,1) TO AA(I,0)-AA(I,2) STEP AA(I,2)
100     Q=S?J;S?J=S?(J+AA(I,2));S?(J+AA(I,2))=Q
110   NEXT J
120 NEXT I
130 ?#E1=0;VTAB 8;HTAB H;PRINT $S$11
140 FOR I=LEN(S) TO 1 STEP -1
150   PAUSE 15;T=S?AA(I,0)
160   FOR J=AA(I,0) TO AA(I,1) STEP -AA(I,2)
170     HTAB H+J;PRINT $T';S?J=CH" ";HTAB H;PRINT $S
180     IF J<>AA(I,1);S?J=S?(J-AA(I,2))
190     PAUSE 15;HTAB H+J;PRINT $11" "
200   NEXT J;S?(J+AA(I,2))=T;HTAB H;PRINT $10$S$11
210 NEXT I;?#E1=#80;VTAB 15
220 END
230 DATA "FRANS.VAN.HOESEL"
```



Voorzitter:
Th. van Kempen.
Het Puyven 71.
5672 RB Nuenen.
Telf 040 - 836210

Secretaris:
A. Jonseling.
Klokkengietershoeve 3
7326 SB Apeldoorn.
Telf 055 - 417314

Penningmeester:
G. Visser.
Haringvliet 391.
8032 HL Zwolle.
Telf 038 - 546561.

Giro 5244293 Bank 93.32.87.283
Beide t.n.v. Penningmeester Acorn computerclub.
Zwolle.

Redactie Acorn Nieuws:
H. Reinders.
Leeuwarderstraat 8
9718 HX Groningen.
Telf 050 - 125458

Bandjesarchief:
P. Grevelt.
Swalmstraat 12
1784 CP Den Helder
Telf 02230 - 37060

Datasheets:
G. Akkermans.
Wikke 1
1273 BR Huizen.
Telf 02152 - 60294.

Drukwerkarchief:
F. Monsato
Biesbosch 153
8032 VD Zwolle.

Uiterste datum inlevering copy:

NR 4. 25 - 5 - 1984.
NR 5. 27 - 7 - 1984.
NR 6. 24 - 8 - 1984.
NR 7. 28 - 9 - 1984.
NR 8. 26 -10 - 1984.
NR 1. 10 -12 - 1984.

DE CLUB-WINKEL:

Geheugenkaart; 16 kByte extra in de ATOM	fl. 50,00
Schakelkaart; meerdere EPROM's op Axxx (incl.74LS133)	fl. 50,00
Programmerkaart; zelf programmeren van EPROMs	fl. XX,XX
Herdruk ACORN NIEUWS 1982; 97 pa. wetenswaardigheden	fl. 6.00
Jaargang 1983; totaal ruim 450 pag.	fl. 30.00
ATOM-WARE deel 1; machinetaal op de ATOM 98 pag.	fl. 6.00

Bestellen:
Bij Uw regionale penning-meester.
Rechtstreeks bij de federatieve penningmeester, dan fl. 5.00 extra voor porto-kosten.
Archiefdiensten:
Voor bandjes, datasheets, EPROMS en alle andere vormen van dienst verlening kunt U terecht bij de regionale archiefdiensten. Mocht in Uw regio iets ontbreken, dan helpt het regionalebestuur U verder op weg.

Voor wat de printen betreft moet enig geduld worden betracht aangezien de kaarten opnieuw worden getekend op eurokaart formaat.

ACORN NIEUWS Uitgave van Fed. Acorn Computerclubs N/B
Verschijnt 8 x per jaar.

De folder van onze computer zei iets in de trand van: DE ACORN ATOM GROEIT MET U MEE! Dat is heel mooi, maar wanneer we als club de hardware van onze computer zonder enig BELEID gaan uitbreiden, lopen we onherroepelijk vast!!. Om hier een klein beetje een lijn in aan te brengen is er een landelijk HARDWARE GROEPJE van 4 mensen ontstaan. We hebben hier nu eens met z'n drieen een zaterdagje aan opgeofferd, om een lijn uit te zetten. In het kort komt het hierop neer:

1. De bus: We hebben met onze computer samen een bussysteem gekocht, net zo als we een nogal afwijkende BASIC gekocht hebben, waar we allemaal? nog steeds mee werken. DIT WORDT DOOR DE CLUB ALS STANDAARD GENOMEN, dus niks gedoe met ELEKTUUR bus, enz.
2. Het BXXX- gebied: Dit gebied is in onze computer gereserveerd voor MEMORY-MAPPED I/O: Hier dus geen RAM!
De connector PL8: Deze blijft in de kast, en is in principe gereserveerd voor BIG BENNY. (B4XX)
Er zal binnenkort een "PL8 extension board" komen, dat buiten de kast nog eens een aantal PL8 connectors uitdekodeerd, die dan op B5XX, B6XX, B7XX, en/of BCXX t/m BFXX komen, dus steeds 256 bytes per connector. Hier kunnen dan I/O zaken als 'sound-generator', 'A/D,en D/A converters', '80 kolomskaart', en andere I/O zaken memory-mapped aan gekoppeld worden.
3. De toekomstige club kaarten zullen zo uitgevoerd worden dat ze zowel in een rack als in de computer kunnen. Verder zullen ze voorzien worden van een DISABLE schakelaar, en een breedte hebben van 19''.
4. Publikatie in acorn nieuws van hardware zaken: Wanneer de redaktie een aantal artikelen toegezonden krijgt, die hetzelfde doel beogen, wordt het ontwerp met de meeste mogelijkheden gepubliceerd.
5. De gedachten voor toekomstige ontwerpen gaan in de richting van:
  - RS 232C serieele interface
  - VDU kaart (80 koloms + 256*512 graphics) (in ontwikkeling)
  - Printerinterface kaart + buffer
  - DOS print (voor elke drive)
  - experimenteerprintjes, voor aan een PL8 connector, waar u zelf uw I/O projecten op kunt bouwen

Om de interresse voor deze dingen te peilen, is elders in dit numer een ENQUETE-FORMULIER opgenomen. GELIEVE DIT IN TE ZENDEN !!!

Rudi van Drunen.

* DE CURSOR nr 3 (BRABANT)
grafisch tekenen van een draak; jaarverslag 1983. Joystick aanpassingen; COS troubles door afwijkende bandsnelheid; ACIA interface op de PL8; Schuivende titels; Orgeltje een uitbreiding van HARPSICHORD; 3D-plot;Vaatdoek;Schakelklok;Delta A/D converter

* DATACHECK nr 3 (DEN HAAG)
ASBK-schakelsysteem ; Catalog van een disk in 3 rijen op het scherm; Spiraal; Chain ; Diskcas schrijft een schijf naar de band;

* DE STACKER mei (NOORD)
statement TLOCK; MAKE voor het maken van een P-CHARME tabel; CLM voor het wissen van geheugen; P-bug; Statement code; COS deel2; 1kByte op #400; Priemgetallen; 200k op een schijf; varkentjes stapelen; grafisch grapje

* ERROR 94 nr2 (NOORD-HOLLAND)
Speech synthesiser voor de ATOM; Computer jargon; zestien bit 6502; Diverse wetenswaardigheden;

* NEWSLETTER nr2 (BELGIE)
4K ram op A000; Rack voor de ATOM; EXPANDIC ATOM BASIC oftewel statement uitbreiding (een beetje achterhaald dus); Een automatische schakelkaart; De ATOM-BASIC COMPILER.; overzicht ATOM ROM routines; De MCDR; Zelf Grafics ontwerpen; De 6522 VIA; De ATOM bus;Bestand organisatie

Federatief Drukwerk Archief

Het federatief drukwerk archief is sinds enige tijd in handen van Fernando Monsato.
Nadat deze het archief heeft geordend kwam hij tot de conclusie dat zeker het laatste jaar niets meer aan het archief was toegevoegd, omdat er niets meer werd toegestuurd door de leden.
Het archief heeft uiteraard alleen zin indien het veelvuldig wordt aangevuld. Derhalve bij deze de oproep :

Stuur alle van belang zijnde drukwerk-informatie aan :
F. Monsate - Biesbosch 153 - 8032 VD Zwolle

Uiteraard kan via het regio drukwerk archief de benodigde informatie voor zover aanwezig hier worden opgevraagt.

A. Jongeling

secretaris F.A.C.C.

In het vorige nummer heeft U de redactionele toespraak moeten missen wegens plaatsgebrek. Op de federatieve vergadering in Utrecht is afgesproken dat bij elke ACORN NIEUWS de programmatuur op diskettes naar alle regio's zal worden gestuurd. Zodoende hoort bij deze ACORN NIEUWS, 2 diskettes met programmatuur. Een overzicht van de tot nu toe verstuurde diskettes volgt nu:

```
Schijf 1 maart 1984.
GCD      1000 AFAF  000E4 002 P-CHARME voorbeeld.
ADV.LDG  2900 AFAF  03D54 003 P-CHARME voorbeeld.
COPY     2900 C2B2  00ADD 041 Diverse copieer programma's
PATTST   2900 C2B2  007D2 04C A.N. 1984 nr. 1 blz. 28
SOMTEST  2900 C2B2  00480 054 A.N. 1984 nr. 1 blz. 91
MJCOS    2900 C2B2  0118C 059 A.N. 1983 nr. 6 blz. 17
DOSSOS   3000 3000  01000 06B A.N. 1984 nr. 1 blz. 59
BASICBB  2900 C2B2  009E0 07B A.N. 1984 nr. 1 blz. 86
P-CHARM  A000 A000  01000 085 foutieve versie.
SOSSRS2  2900 C2B2  02AA8 095 A.N. 1984 nr. 1 blz. 59
ASN      2900 C2B2  01691 0C0 A.N. 1984 nr. 1 blz. 22
RINORDS  2900 C2B2  00CEB 0D7 A.N. 1984 nr. 1 blz. 10
SECTOR   2900 C2B2  00BB1 0E4 wordt vervangen op juni schijf.
SRCMAKE  2900 C2B2  010AB 0F2 A.N. 1983 nr. 6 blz. 74
DB2.0    8200 8DD9  00EEA 103 A.N. 1983 nr. 6 blz. 90
TURTLE   2900 2900  02700 112 A.N. 1983 nr. 5 blz. 11
PANIC    2900 3C00  01344 139 toegevoegd spelletje
PAINTER  2900 3C00  01344 14D toegevoegd spelletje
HYPERFI  2900 CE86  01300 161 toegevoegd spelletje
SDSOBJ2  1800 1800  00800 174 A.N. 1984 nr. 1 blz. 59

Schijf 1 mei 1984.
SDRAWOB  9C00 9C00  01400 002 A.N. 1984 nr. 1 blz. 41 (object.)
PALET    2200 0000  06000 016 A.N. 1984 nr. 1 blz. 41 (tekening)
SDRAWSR  2200 AFAF  05B5D 076 A.N. 1984 nr. 1 blz. 41 (source)
3-DPLOT  2200 0000  06000 0D2 A.N. 1984 nr. 1 blz. 41 (tekening).
SPRDATA  1C00 C2B2  00400 132 A.N. 1984 nr. 1 blz. 41 (text-data)
ASS-R00  A000 C2B2  01000 136 A.N. 1984 nr. 2 blz. 65 (object)
P-CHARM  A000 C2B2  01000 146 vervallen versie
ZREFSEA  2900 AFAF  00CA8 156 A.N. 1984 nr. 3 blz. 45
ZEROREL  2900 AFAF  005D5 163 A.N. 1984 nr. 1 blz. 64
BUFFER   2900 AFAF  005D5 169 A.N. 1984 nr. 3 blz. 92
INFOMAS  8200 8200  01DBA 16F A.N. 1984 nr. 2 blz. 10

Schijf 2 mei 1984.
ASSEMSR  2900 AFAF  035BE 002 A.N. 1984 nr. 2 blz. 65 (source)
CASCAT   2900 C2B2  004F5 038 A.N. 1984 nr. 2 blz. 75
MOREDIR  2900 AFAF  00980 03D A.N. 1984 nr. 2 blz. 38
MEMEDIT  2900 C2B2  00EDE 047 A.N. 1984 nr. 2 blz. 71
PROMMER  2900 C2B2  00AFF 056 A.N. 1983 nr. 6 blz. 26
GFILL    2900 AFAF  00FA9 061 A.N. 1984 nr. 2 blz. 62
ATTACH   2900 AFAF  001D3 071 A.N. 1984 nr. 2 blz. 51
EXTERN   2900 AFAF  003CF 073 A.N. 1984 nr. 2 blz. 51
VLIST    2900 AFAF  002C2 077 A.N. 1984 nr. 2 blz. 50
DIS      2900 AFAF  00611 07A A.N. 1984 nr. 2 blz. 52
CODE     2900 AFAF  004DB 081 A.N. 1984 nr. 3 blz. 91
SHAPEPL  8200 C2B2  0100F 086 A.N. 1984 nr. 2 blz. 58
PLOTDEM  8200 C2B2  00C55 097 A.N. 1984 nr. 2 blz. 58
```

MEMORIE	2900	C2B2	00862	0A4	A.N. 1984 nr. 2 blz. 93
SOUND	2900	AFAF	0071E	0AD	A.N. 1984 nr. 2 blz. 95
DOOLHOF	2900	C2B2	005EE	0B5	A.N. 1984 nr. 2 blz. 97
ATOMMAN	2800	C2B2	01400	0BB	toegevoegd spelletje
TORPEDO	2900	3C00	0130B	0CF	toegevoegd spelletje
3DASTER	2900	C2B2	01300	0E3	toegevoegd spelletje
PLAKOTO	2900	C2B2	01100	0F6	toegevoegd spelletje
BACKGAM	2900	C2B2	011A3	107	toegevoegd spelletje
DAMMEN1	8200	C2B2	0194B	119	toegevoegd spelletje
DAMMEN2	2C00	3500	01200	133	toegevoegd spelletje

Schijf 1 juni 1984.

sys-txt	2800	A071	03012	002	A.N. 1984 nr. 3 blz. 68
str-txt	2800	A071	00D98	033	A.N. 1984 nr. 3 blz. 68
ASBK=i	A000	A000	00FFF	04B	A.N. 1984 nr. 3 blz. 68
ASBK-S	8200	C2B2	01BC4	05B	A.N. 1984 nr. 3 blz. 68
FUNCTIE	8200	C2B2	016FB	077	A.N. 1984 nr. 3 blz. 68
asb-box	2800	A071	053CB	08E	A.N. 1984 nr. 3 blz. 68
restxt	2800	A071	017F6	0E2	A.N. 1984 nr. 3 blz. 68
stattxt	2800	A071	01278	0FA	A.N. 1984 nr. 3 blz. 68
P-CH173	A000	A000	01000	10D	A.N. 1984 nr. 3 blz. 67

Schijf 2 juni 1984.

PCH-173	A000	A000	01000	002	A.N. 1984 nr. 3 blz. 67
SECTOR	2900	AFAF	00C37	012	A.N. 1984 nr. 3 blz. 21
PYTHAGO	2200	0000	06000	01F	tekening superdraw.
SORT	8200	AFAF	00951	07F	A.N. 1984 nr. 3 blz. 30
MATRIX	2900	C2B2	005A0	089	A.N. 1984 nr. 3 blz. 90
B-PRINT	2900	AFAF	003A4	08F	A.N. 1984 nr. 3 blz. 92
CALCUL	2900	AFAF	0089F	093	A.N. 1984 nr. 3 blz. 83
INTERP	2900	AFAF	005ED	09C	A.N. 1984 nr. 3 blz. 36
TELMACH	2900	AFAF	003D4	0A2	
SPLITSC	2900	AFAF	0161B	0A6	A.N. 1984 nr. 3 blz. 87
MAGVIE	2900	C2B2	00ABA	0BD	A.N. 1984 nr. 3 blz. 38
ZOMBIE	2900	AFAF	00D74	0C8	A.N. 1984 nr. 3 blz. 80
HEXIT	2900	AFAF	01634	0D6	A.N. 1984 nr. 3 blz. 58
HEXDATA	8200	AFAF	0024E	0ED	A.N. 1984 nr. 3 blz. 58
VDU80AS	2900	AFAF	0331C	0F0	A.N. 1984 nr. 3 blz. 41
DATAST	2800	C2B2	01859	124	A.N. 1984 nr. 3 blz. 50
SORTSRC	2900	AFAF	0043B	13D	A.N. 1984 nr. 3 blz. 50
VERDEEL	2900	AFAF	0029B	142	A.N. 1984 nr. 3 blz. 49

Op vele regiobijeenkomsten wordt reeds druk gecopieerd, zadat een bezoekje daaraan waardevolle programmatuur opleverd. De schijven die teruggestuurd worden aan de redactie worden benut voor de toelevering van copy. Dus textfile's en programmatuur kunt u inleveren bij Uw regio.
Hartelijk dank ook voor de overwegend positieve reactie's die wij mochten ontvangen. Degene die ACORN NIEUWS te moeilijk vinden worden bij deze aangemoedigd om zelf eens ietsin te sturen.

De redaktie van ACORN NIEUWS.
Rudy van Drunen.
Frans van Hoesel.
Hans Marks.
Henk Reinders.

2 JUNI 1984 12.30 UUR	NOORD-H	DE KOPEREN KNOOP LIMBURG VAN STIRUMSTRAAT AMSTERDAM.
4 JUNI 1984 20.00 UUR	ARNHEM	t.o. POSTGIRO VELPERWEG 56-A (kelder) ARNHEM.
5 JUNI 1984 20.00 UUR	DELFT	E-KAFE GEBOUW ELEKTRO TH WIJK DELFT.
6 JUNI 1984 20.00 uur	CENTRUM	DE LANTAERN UTRECHTSESTRAATWEG 6 NIEUWEGEIN.
7 JUNI 1984 20.30 UUR	BRABANT	GEBOUW B.R.A.N. HEVEL 7 VORSTENBOSCH.
8 JUNI 1984 20.00 UUR	DEN HAAG	EXODUSKERK BERESTEINLAAN 263 DEN HAAG.
9 JUNI 1984 10.30 UUR	BELGIE	OXACO-CEPENBERGCENTER BORSBEEKSE STEENWEG 45 BOECHOUT (bij HOVE).
13 JUNI 1984 20.00 UUR	TWENTE	MEPA-GEBOUW WIERDENSESTRAAT 40 ALMELO
14 JUNI 1984 19.30 UUR	NOORD	MUNSTERHOES MUSTERHEERD. GRONINGEN.
19 JUNI 1984 20.00 UUR	BRABANT	DE WERFF V/D WERFFSTRAAT EINDHOVEN.
20 JUNI 1984 20.00 UUR	NOORD	MARTIN VAN VLISSINGEN. ACHTER DE GROTE KERK 25 LEEUWARDEN.
23 JUNI 1984 14.00 UUR	OV/GELD	WIJKCENTRUM DE BOLDER. DOBBE 62 (WIJKCENTRUM A-LANDEN) ZWOLLE.
25 JUNI 1984 20.00 UUR	ROTTERD	SPEELTUINGEBOUW HET WESTEN SPAANSEWEG 38 ROTTERDAM-WEST.
5 JULI 1984 20.30 UUR	BRABANT	GEBOUW B.R.A.N. HEVEL 7 VORSTENBOSCH.

We vallen direkt met de deur in huis:
EXEC-files (voor DOS-gebruikers) kunnen worden gebruikt om een hoop typwerk te besparen. Deze files kunnen namelijk worden gevuld met commando's die anders vanaf het toetsenbord ingetikt zouden moeten worden. Een probleem is hierbij echter het aanmaken van zo'n file. Het volgende programma vereenvoudigt die opgave aanmerkelijk:

```
 10 PROGRAM CREATE EXEC-FILE
 20
 30 DIM F20,A20
 40 INPUT "FILENAME "$F
 50 IF NOT FIN$F GOTO 100
 60   PRINT "FILE BESTAAT REEDS"'
 70   INPUT "OVERSCHRIJVEN (J/N) "$A
 80   IF ?A=CH"N" GOTO 40
 90   SHUT 0
100 Q=FOUT$F
110 INPUT "TEXTPAGE "P
120 DO
130   SPUT Q,$P
140   PRINT $P'
150   P=P+3+LEN P
160 UNTIL P?-2)127 ;REM EINDE TEKST ?
170 SHUT Q
180 END
```

Op regel 50 wordt gecontroleerd of de file reeds bestaat, door hem (onschuldig) te openen als random input file. Is de file-handler gelijk aan nul, dan bestaat die file nog niet en wordt er gesprongen naar regel 100. Bovenstaand programma kan bv. geladen worden op #8200. Op #2900 kan dan een (BASIC) tekst staan die de verschillende commando's bevat. Voorbeeld:

```
10 *CAT
20 *MON
30 LIST 10,50
40 /PRINT/P./
50 /GOSUB/GOS./
60 /RETURN/R./
70 LIST 10,50
```

Dit is natuurlijk een onzinnig voorbeeld, maar dat mag de pret niet drukken. Het programma dat de EXEC-file maakt kan nu gestart worden door het intikken van "CREATE". Het programma vraagt dan om een file-name, die u dan braaf intikt. Vervolgens komt een lastige vraag waarop u antwoordt met " #2900". Het 'programma' dat staat op #2900 wordt dan zonder regelnummers op schijf gezet, waarna het kan worden uitgevoerd met:

```
*EXEC file-name
```

LET OP: alles wat normaal vanaf het toetsenbord zou worden ingelezen, komt nu van schijf. Hierdoor werken INPUT-statements ook niet meer met het toetsenbord. Wel werkt het INKEY-statement omdat dit rechtstreeks het toetsenbord scant. Als alle regels uit de file geweest zijn komt de promt weer terug en kan weer vrolijk verder gespeeld worden.
p.s. Werkt niet samen me CHARON.

Wanneer we de DOS-rom bestuderen zien we dat de controler een IC van het type 8271 de computer een hoop werk uit handen haalt. In het COS systeem wordt elk bitje door de prosessor naar de band gestuurd. (zie elders in dit nummer). In het DOS systeem geven we de controler een compleet byte en vertellen hem waar deze op de schijf gezet moet worden.
De schijf is namelijk geformatteerd in 40 tracks en 10 sectoren. Dit wil zeggen dat op de schijf 40 cirkels zijn gevormd en dat elke cirkel is onderverdeeld in 10 deeltjes. Een strenge orde in het opbergen is dus gewenst.
Behalve het saven kent de controler nog veel meer opdrachten. Deze opdrachten moeten vergezeld gaan van parameters. Dit kun je vergelijken met een Mnemonic in de machinetaal welke uit een opcode en 0,1 of 2 operanden bestaat. Zo ook met de opdrachten voor de controler. Wanneer we de controler een opdracht tot saven geven behoren hierbij drie parameters. En wel:
a. op welke track moet er gesaved worden.
b. op welke sector moet begonnen worden te saven.
c. hoeveel sectoren moeten er gesaved worden.

Een sector bestaat in het ATOM systeem uit 256 bytes. Wanneer de opdracht met de parameters aan de controler zijn aangeboden, begint de controler interrupts af te geven. Dit zijn NMI interrupts, wat inhoud dat deze niet kunnen worden genegeerd. De controler geeft nu voor elk byte wat hij moet saven een interrupt af. Bij ieder interrupt wordt de interrupt routine afgehandeld waarbij een byte uit het geheugen wordt gehaald en aan de controler wordt aangeboden. Dit gaat net zolang door totdat het aantal sectoren is afgewerkt. Tot slot wordt er nog een extra interrupt afgegeven waarbij een vlaggetje in een register wordt opgezet. Indien er een comminucatie stoornis tussen drive en controler optreed of de drive de zaak niet kan verwerken wordt dit vlaggetje ook opgezet. Dit register wordt elke keer in de interupt routine getest. Wanneer deze dus gezet is moet er geen data aan de controler worden aangeboden. Inplaats daarvan moet het resultaat register worden bekeken. Indien hier alles OK is werdt het vlaggetje gezet omdat de controler klaar was met zijn werk. Indien er iets loos is staat in dit register een code welke wij beter als een DISK ERROR kennen. Het DISK ERROR nummer wordt dus door de controler gegeven. Dit heeft dus niets te maken met een software-matige BRK welke verantwoordelijk is voor het normale error-nummer.

Dit zijn de disk error codes. Een aantal zijn ongedefinieerd.

00 alles is OK.
02 SCAN MET EQUAL
04 SCAN MET NOT EQUAL
06 -----
08 CLOCK ERROR
0A LATE DMA
0C ID CRC ERROR
0E DATA CRC ERROR
10 DRIVE NOT READY
12 WRITE PROTECT
14 TRACK ZERO NOT FOUND
16 WRITE FAULT

18 SECTOR NOT FOUND
1A -----
1C -----
1E -----

De ATOM-DOS gebruikt maar een aantal intruktie's van de controler. De eerste twee bits van de opdracht geven aan voor welke drive de opdracht bestemd is. Dan nu de gebruikte opdrachten met hub code's voor drive Ø:
35 specificatie commando
2C read drive status.
3A write special register.
23 formateren (alleen in format-programma)
13 read data
ØB write data

Niet gebruikte opdrachten zijn:
29 seek command
3D read special register.
1B read id command
17 read data en delete data
ØF write deleted data
1F verify data and deleted data
ØØ scan data
Ø4 scan data en deleted data

Hierbij is uigegaan van het variable lenght/multi record formaat, waarbij bij het formatteren kan worden aangegeven hoe groot een sector moet zijn. In het geval van ons DOS systeem is een sector 256 bytes.
Het 128 bytes single formaat kan door ons niet gebruikt worden en daarom besteden we daar ook verder geen aandacht aan.

Ik heb expres in deze uiteenzetting jullie niet met adressen om de oren willen slingeren. Dat doen we eventueel een volgende maal. De reden hiervan is dat de DOS voor het grootste deel in zijn programma aan het rekenen. Dit rekenen doet hij aan de hand van zijn katalogus. Hierin staan alle gegevens voor de DOS waarbij hy tot de slotsom kan komen: hoeveel sectoren; op welke track en op welke sector beginnen.

FORTH

In deze aflevering van de Forth serie een praktisch programma: een decompiler. Een decompiler is een soort disassembler, welke echter geen machinetaal disassembleert maar een Forth woord decompileert. Dit laat zich het beste toelichten aan de hand van een klein voorbeeld programma. Stel U voor we definiëren zelf een nieuw Forth woord genaamd: "TITLESCREEN" dat dienst doet als welkomstpagina bij een door ons geschreven programma.

```
: TITLESCREEN
    12 EMIT
    CR CR CR
    ." *** WELCOME TO USERPROGRAM ***"
    CR CR
    ."      HIT A KEY TO CONTINUE"
    KEY DROP
;
```

Na een aantal weken willen we nog eens bekijken hoe we het woord TITLESCREEN ook al weer gedefiniëerd hadden. Dat kan natuurlijk makkelijk als we een listing of iets dergelijks hebben. Is dit echter niet het geval dan kunnen we mooi gebruik maken van het hier gepresenteerde programma door in te typen:

```
DECOMPIL TITLESCREEN<CR>
```

en onderstaande listing verschijnt op het scherm.

```
[1]  [2]  [3]   [4]
  1 XXXX XXXX ( LIT )
  2 XXXX XXXX 12
  3 XXXX XXXX EMIT
  4 XXXX XXXX CR
  5 XXXX XXXX CR
  6 XXXX XXXX CR
  7 XXXX XXXX ." *** WELCOME TO USERPROGRAM ***"
  8 XXXX XXXX CR
  9 XXXX XXXX CR
 10 XXXX XXXX ."      HIT A KEY TO CONTINUE"
 11 XXXX XXXX KEY
 12 XXXX XXXX DROP
 13 XXXX XXXX ;S
```

Als algemene regel kunnen we aannemen: Alles wat in kolom [4] staat kunnen we, als het niet tussen ronde haken staat, letterlijk intoetsen om de definitie te reconstrueren. Waarden die tussen haken staan moeten we verder bekijken en vervangen door de bijbehorende conditionals, zoals bijv. IF...ELSE...THEN. De enige uitzondering hierop vormen de vormen (LIT) en (CLIT) deze kunnen we gewoon weglaten omdat het systeem ze zelf genereert als we een numerieke waarde intoetsen.

De betekenis van de verschillende kolommen in de decompiler listing is als volgt:

[1]: serie nummer van de gedecompileerde code.
[2]: adres van de code [HEX].
[3]: hexadecimale representatie van de code.
[4]: source code. (numerieke waarden worden weergegeven in het decimale stelsel.

Als U de decompiler intoetst in een niet aangepast Forth systeem (zie ook de vorige afleveringen van deze serie) dient U DP en FENCE te wijzigen in #8200 in plaats van #8C00, omdat de EDITOR en de DECOMPILER samen te groot zijn.
Als het programma runt kunt U het onderbreken door <ESC> te toetsen. Als U hierna <CR> toetst zal de decompilatie afgebroken worden, elke andere toets zal een hervatting van de decompilatie inhouden. Door de ESC-toets in te houden krijgt U een soort single step mogelijkheid.

Bij het gebruik van de decompiler zijn een aantal foutmeldingen mogelijk. Deze zijn in onderstaand lijstje samengevat.

Foutmeldingen:
----

9 : Geen ":" definitie.
10 : Machine code functie.
11 : Variable.
12 : Constant.
13 : User variable.
14 : Vocabulary/<BUILDS..DOES>

Indien U een van deze meldingen krijgt wil dit zeggen dat de decompiler dit type definities niet aan kan.

Bron: PCW october 1983.

```
BASE @
FORTH DEFINITIONS HEX
: CHECK
    [COMPILE] ' DUP CFA DUP @
    DUP 2D87 = B ?ERROR
    DUP 2D6B = C ?ERROR
    DUP 2DA0 = D ?ERROR
    DUP 317A = E ?ERROR
    2DUP 2 - = A ?ERROR
    2D2F XOR 9 ?ERROR
    DROP
;

0 VARIABLE NCODE 0 VARIABLE BOFS
0 VARIABLE +OFS  0 VARIABLE FLAG
4 CONSTANT FIELD 2 CONSTANT RIGHT

: OFLAG 0 FLAG ! ;
: TAB RIGHT SPACES ;
: DSIG
    DECIMAL FIELD .R HEX TAB
;
: UNSIG 0 FIELD D.R TAB ;
: NPRINT
    NCODE @ 1+ DUP
    CR DSIG NCODE !
;

: LPRINT
    NPRINT
    OVER 2+ DUP UNSIG
    @ DUP UNSIG
    DECIMAL . HEX
    2 +OFS +!
    OFLAG
;
```

```
: ID1
   DUP 2828 = IF
   ." ( LIT )"
   LPRINT
   THEN
;

: QUOTE
   2E EMIT 22 EMIT 20 EMIT
   COUNT TYPE
   22 EMIT
;

: ID2
   DUP 31E4 = IF
   OVER 2+ DUP QUOTE
   C@ 1+ +OFS +!
   OFLAG
   THEN
;

: ID3
   DUP 2910 = IF
   ." DO"
   OFLAG
   THEN
;

: ID4
   DUP 288A =
   IF ." LOOP"
    2 +OFS +! OFLAG
   ELSE DUP 28E0 =
    IF ." +LOOP"
     2 +OFS +! OFLAG
    THEN
   THEN
;

: ID5
   DUP 288D =
   IF ." ( BRANCH" LPRINT
     ." )"
   ELSE DUP 28A2 =
    IF ." ( OBRANCH"
     LPRINT ." )"
    THEN
   THEN
;

: ID6
   DUP 285B =
   IF ." ( CLIT )"
    NPRINT OVER 2+
    DUP UNSIG
    C@ DUP FIELD
    .R TAB DECIMAL
    . HEX 1 +OFS +!
    OFLAG
   THEN
;

: IDE
   DUP 30D3 =
   IF ." COMPILE" DROP
    NPRINT 2+ DUP DUP
    UNSIG @ DUP
    UNSIG 2 +OFS +!
    1 FLAG !
   THEN
;

CREATE VALID 2A9 , 6320 ,
    A528 , 3892 , 3E9 , 9285 ,
    2B0 , 93C6 , 8EB6 , A2 ,
    92B1 , 3910 , 92C6 , 92A5 ,
    FFC9 , 2D0  , 93C6 , E0E8 ,
    F020 , B12A , 1092 , 29ED ,
    861F , C594 , 9094 , 911E ,
    A490 , B194 , 2992 , 917F ,
    8890 , F7D0 , 8EA6 , CACA ,
    92A5 , 95   , 93A5 , 195  ,
    1A9  , 214C , A62B , 988E ,
    FBF0 ,
SMUDGE

: NFA= DROP NFA OVER = ;

: PRINT
   DUP DUP
   C@ 40 AND
   IF ." [COMPILE] "
   THEN 284B = FLAG @
   XOR 3 =
   IF ." ; " DROP
   ELSE ID.
   THEN
;

: [D]
   DUP @ ." [ "
   DECIMAL . HEX
   ." , ]"
;

: IDF
   FLAG @
   IF HERE VALID
    IF HERE LATEST [ 2955 , ]
     IF NFA=
      IF PRINT
      ELSE DROP [D]
      THEN
     ELSE DROP [D]
     THEN
    ELSE [D]
    THEN
   ELSE DROP
   THEN
;
```

```
: LAST?
   2R50 = DUP
   IF FLAG @ AND 1 XOR
   THEN
;

: DECOMPIL
    BASE @ CURRENT @
    CHECK
    DEFINITIONS
    0 NCODE ! 0 BOFS !
    BEGIN
     0 +OFS ! 2 FLAG !
     NPRINT
     DUP
     BOFS @ + DUP DUP UNSIG
      @ DUP UNSIG
      ID1 ID2 ID3 ID4 ID5 ID6
      NOOP NOOP
      IDE IDF
      +OFS @ 2+ BOFS +!
      @ LAST?
      ?ESC DUP
      IF DROP KEY D =
      THEN OR
    UNTIL
    DROP
    CURRENT ! BASE !
;

BASE !
```

«*»

```
  0 REM LIST
 10
 20 REM KOMMANDO-TRACER 24-3-84
 30 REM ROB VAN DORT
 40 REM GEBRUIKT JOSBOX; 'LINK 2' MOET AANWEZIG ZIJN
 50 REM HET KOMMANDO ACHTER DE REM OP DE EERSTE REGEL
 60 REM WORDT GETRACEERD
 70 REM HOUDT precies HET FORMAAT VAN DE EERSTE REGEL AAN :
 80 REM REGELNR., SPATIE, REM, SPATIE, (KOMMANDO)
 90
100 S=?18*256+8;REM STARTADRES KOMMANDO NAAM
110 DIM LL(3),P(-1)
120 PRINT $21
130[
140:LL0
150 LDX@0
160:LL1
170 LDA S,X;STA #100,X
180 INX;CMP@13;BNE LL1
190 JMP #C2D5 \interpreteer commando op #100
200]
210 PRINT $6$12
220 $#100="STEP LL0,#C309"
230 LINK#C2D5
240 'END'
250
260 DE DISPLAY-PARAMETER (,#C309) VOORKOMT DAT DE INTERPRETER
270 U EEN LANGE SAAIE LUS VOORSCHOTELT
280 MAAR KAN EVENTUEEL WEGGELATEN WORDEN
```

MEER INFOMASTER

Nieuwe ontwikkelingen aan en wetenswaardigheden van het Info-master-terrein

Naar aanleiding van vragen van verschillende gebruikers is Arie Marchal weer in zijn toverdoos gedoken en heeft enkele aanvullingen voor het infomaster programma geschreven:
1) overbrengen van een bestand, aangemaakt met INFO82(L) naar Infomaster
2) formatteren (=corrigeren) van de scherm- en/of printer-output
3) het OPTELLEN van de waarde die een bepaald item heeft in alle aanwezige records
4) enkele hulpmiddelen

1 Overbrengen van databestanden van INFO82(L) naar Infomaster
Vroege gebruikers van eerdere versies van Infomaster, tw INFO82 of INFO82L, hebben wellicht de behoefte het databestand over te brengen naar Infomaster, zonder dit opnieuw helemaal met de hand te moeten invoeren.
Doordat in Info82(L) VASTE geheugengebieden werden gebruikt voor de 'record'- en 'ascii'-file, terwijl dit in Infomaster een aaneengesloten verschuivend blok is, moest een apart programma worden geschreven.
Daartoe wordt Infomaster geladen:
- regel 714 en verder verwijderen
- toevoegen regel 1000 - 1100 volgens programma onder no.????? opgenomen in het drukwerkarchief
- laad nu het bestand door 2 maal *LOAD"filenaam" in casu voor de 'record'- en 'ascii'-file
- run infomaster en geef bij de vraag KEUS?)(escape)
- type in G.1000;(return) en wacht op de tekst
- verplaats de ASCII file zoals wordt opgegeven door het programma mbv 'reloc' of 'copy' uit de Josbox of een equivalent daarvan
- laad het oorspronkelijke Infomaster-programma.

2 Het formatteren van de output
Formatteren of vormgeven van de output van Infomaster, is voor allerlei toepassingen interessant. Bijvoorbeeld:
- items verdelen over meerdere regels
- bij verdeling over meerdere regels de 2e regel laten inspringen
- printen van etiketten ed

Items verdelen over meerdere regels
Wil men bv item 1 t/m 4 op de eerste regel en item 5 t/m 8 op de tweede regel of alle items op een afzonderlijke regel, dan is dat mogelijk door de navolgende regels aan het Infomaster-programma toe te voegen.
regel 300 : toevoegen ;K=8
regel 303 : 5 wordt 6
regel 309 : L.S wordt VALS
regel 567 : label x weghalen
Toevoegen:
regel 280 IFI=5P."LF"

regel 565xIFP?J&1E;P.';U?1=U?1+I;C=0
Bij keuze 4 uit het printmenu verschijnt nu tav ieder item ook de vraag 'LF'. Bij N gaat alles zijn oude gang. Bij J verschijnt het desbetreffende item op een nieuwe regel en de daarna volgende lopen daar gewoon achteraan.

Inspringen bij verdeling over meerdere regels
Voor de overzichtelijkheid van tekst kan het aantrekkelijk zijn op de tweede regel van een record in te springen, ofwel ergens in een regel een 'open-vlak' te creeeren. Dit kan bereikt worden door bij de eerste aanmaak van de database een tweetal items toe te voegen die als 'naam' een 'spatie' krijgen.
Bij het opmaken van de output, worden dan mbv 'keuze 6' uit het printmenu, deze lege items op de juiste plaats gemanouvreerd en vervolgens bij 'keuze 4' uit het printmenu voorzien van de gewenste regellengte, waarmee de lengte van het inspringen wordt bepaald. Natuurlijk heeft deze werkwijze het nadeel dat bij het vullen van de database, deze 'lege' items verschijnen, maar met het speciale commando kan daar gemakkelijk 'overheen' gesprongen worden.

Het printen van etiketten
Als de voorgaande werkwijzen is gevlgd, is het ook mogelijk om daadwerkelijk etiketten te printen. Er kunnen immers een aantal 'lege regels' worden gegenereerd cq het juiste beginpunt voor het volgende record.
Men voegt dan mbv 'keuze 6' van ht printmenu een of meerdere lege items achter elkaar, en geeft met 'keuze 4' van het printmenu bij 'LF' J
Voor degene die reeds Infomaster gebruikt is er dan nu het volgende probleem, om alsnog (lege) items aan bestaande bestanden toe te voegen. Geduld, er is al een hulpprogramma door Arie gemaakt, maar er zit nog een hardnekkige 'bug' in. Zodra deze eruit is wordt U geinformeerd.

Het optellen van itemwaarden
Voor diegenen die infomaster toepassen op een magazijnbestand, kan het handig zijn de waarden die een bepaald item in alle records heeft, te kunnen optellen.
- verwijder regel 714 en volgende
- voeg toe regel 714 t/m 754 volgens onderstaandu listing

```
714GOS.690;IN.''"WELK ITEMNUMMER OPTELLEN"$S;IFL.S(1;G.a
724J=VALS;IFJ)A;G.714
734X=0;T=RR0;DOR=T+O?J;GOS.q
744X=X+D;T=T+B;U.T)=RR1
754P.'"TOTAAL "$NNJ" = "X';G.a
```

- RUN infomaster en kies "merge"(7)

Ofwel:
Verwijder de keuze 8 uit infomaster, kort een paar woorden wat in, geef aan de "optelroutine" een hoger regelnummer, dan kan deze routine nog net erbij gevoegd worden (met bv.keuze 8 of 9)

4. Enkele wetenswaardigheden:

a. Verkort Infomasterprogramma

Voor diegenen die een bestaand databestand willen "gebruiken", of door anderen laten gebruiken (bv. alleen sorteren, selecteren, opzoeken en afdrukken) dus zonder het bestaande te veranderen, kunnen evt. gebruik maken van een verkorte versie van Infomaster, die precies past in 4K, dus in een EPROM.
Deze verkorte versie heeft voorasnog helaas het nadeel dat de "merge"-faciliteit ontbreekt.
Indien hiervoor belangstelling bestaat zal ik deze versie in het bandjesarchief laten opnemen. (B.T.)

b. Ten behoeve van de echte "knutselaar" is in het drukwerkarchief onder no. a116 beschikbaar een overzicht van alle in het Infimasterprogramma gebruikte variabelen (adres, naam, omschrijving, voorbeeld van inhoud), dit als service van Arie voor diegenen die zelf willen sleutelen aan het programma.

Een reaktie van FRANK DE GROOT op het prgramma gezichtsbedrog van Dhr. Dekker.

```
 10 REM GEZICHTSBEDROG
 20 REM FRANK DE GROOT
 30 GRMOD;P.'"          GEZICHTSBEDROG"''''''''''''
 40 MOVE 71,167;DRAW 185,167
 50 P."      DE VERTIKALE LIJNEN"'"      LOPEN EVENWIJDIG!"
 60 FOR X=50 TO 200 STEP 30
 70   MOVE X,40;DRAW X,150
 80 N.
 90 FOR X=50 TO 200 STEP 60
100   FOR Y=45 TO 145 STEP 15
110     MOVE (X-6),Y;DRAW (X+6),(Y+12)
120   N.
130 N.
140 FOR X=80 TO 200 STEP 60
150   FORY=45 TO 145 STEP 15
160     MOVE(X+6),Y;DRAW (X-6),(Y+12)
170   N.
180 N.
190 END
```

BBC - BASIC

Zoals U wellicht weet, is er voor de Atom een BBC-BASIC-kaart beschikbaar, waarmee de Atom BBC-Basic kan praten. Let wel: De Atom wordt niet veranderd in een BBC, alleen de BBC-Basic is geimplementeerd. BBC-Basic lijkt veel meer op Microsoft Basic dan Atom Basic (ook met P-CHARME erin), maar bevat tevens een aantal uitbreidingen t.o.v. Microsoft Basic, zoals procedures en functies met locale variabelen. Over variabelen gesproken: deze mogen willekeurige lengte hebben; we hebben dus de beschikking over lange variabele-namen. Verder bevat BBC-Basic een ingebouwde met Basic mengbare assembler, net als in Atom Basic, maar deze assembler is, doordat BBC-Basic lange variabelen toelaat, symbolisch. Verder kent BBC-Basic de ? en ! operatoren, het werken met hex getallen, meerdimensionale array's, Microsoft-achtige stringhandling enz.,enz., te veel om op deze plek op te noemen.
Nog afgezien van de prijs (+F.300,-) heeft deze BBC-kaart een aantal nadelen, zoals de zeer ongunstige plaatsing in de kast, waardoor een eventuele geheugen- of schakelkaart en gestapelde RAM's verwijderd moeten worden.
Maar deze kaart biedt wel een zeer luxe, complete, uitgebreide en snelle (sneller dan Atom Basic) Basic interpreter. Toch vinden wij de voordelen niet opwegen tegen de nadelen.
Maar ... het moet toch mogelijk zijn de inhoud van de ROM's op deze kaart (verkregen met de medewerking van een BBC-kaart-bezitter) te kopieren en in het RAM-geheugen van de Atom in te laden? Dan kan ik toch BBC-Basic gebruiken?
In principe juist, maar op het BBC-kaartje wordt in BBC-mode de gehele memory-map door elkaar gegooid. Op het kaartje zit o.a. 20K ROM: 16K BBC-Basic interpreter en 4K operating system (zg. MOS-ROM). Deze laatste 4K bevindt zich op adres #F000-#FFFF en de eerste 16K bevindt zich op #8000-#BFFF. Ook wordt het RAM-geheugen verplaatst: beeldscherm naar #4000-#57FF, text-ruimte naar #0800-#1BFF, werkgebieden naar #0000-#07FF. De I/O bevindt zich nu op #7000-#7FFF: PPI(8255) op #7000 en VIA(6522) op #7800. U ziet, dit lukt niet zomaar vanzelf.
Om toch BBC-Basic te draaien moet je dus de complete 20K objectcode uit de ROM's met de hand gaan relocaten naar (in een 'normale' uitgebreide Atom) bruikbare adressen. Dit is een gigantisch karwei. Niet alleen de absolute adressen (deze zijn, hoewel talrijk nog het gemakkelijkst aan te passen) maar ook tabellen met adressen, beginwaarden van pointers enz. moeten worden aangepast. Indien U niet ziet wat de problemen zijn, raad ik U aan bv. de Josbox te relocaten naar bv. #7000. De Josbox is 'slechts' 4K, terwijl we hier spreken over 20K.
Na lang intern beraad hebben we toch de knoop doorgehakt om wel te gaan relocaten en op dit moment hebben Will Verhoeven (nu in Amerika) en ondergetekende de BBC-Basic op papier gerelocated.
Wat betreft de geheugenindeling en de benodigde hardware het volgende:
-De 16K BBC-Basic interpreter komt op #4000-#7FFF, bv. in de geheugenkaart.
-De 4K MOS-ROM komt op #A000-#AFFF, bv. in RAM of in EPROM op een schakelkaart of gewoon in de IC24-voet in EPROM.
-Het BBC-werkgebied #400-#800 komt nu vanaf #2000. Hier moet U dus RAM hebben.(bv. door stapelen te verkrijgen).

-Verder moet U uiteraard op #3C00-#4000 RAM hebben (bv. stapelen). Samengevat: minimaal nodig: RAM op #2000-#7FFF en liefst RAM op #AXXX. Verder uiteraard een VIA (maar die had U toch al) voor het TIME-statement.
Het eigenlijke BBC-Basic programma kan dan op #2400-#3FFF en/of op #8200-#97FF (of #9FFF bij gestapeld hoog geheugen). Dit lijkt niet zo gek veel, maar bedenk dat de opslag van programma's zeer efficient en compact gaat m.b.v. tokens. Trouwens, ook met het BBC-kaartje kunt U geen gebruik maken van Uw geheugenkaart, en een BBC-B computer heeft in MODE 0 ook nog maar 8K RAM vrij voor programma's.
Het voordeel van het op deze wijze beschikbaar maken van BBC-Basic boven de aanschaf van een BBC-kaart is dat dit gratis is, en (veel belangrijker) dat in een klap een groot aantal mensen BBC-Basic kunnen draaien.
Tijdens het relocaten zijn we een groot aantal bytes uit de categorie 'twijfelgeval' tegengekomen, waarvan we niet weten wat die betekenen en of we die dus moeten aanpassen of niet. Dit zal nog een hele uitzoekerij worden. Op dit moment ben ik bezig met het invoeren van alle wijzigingen. Dit zijn er nogal wat. Mocht dit monniken-project ooit nog af komen, dan hoort U daar zeker nog van. Wilt U meer weten, neem dan even contact met mij op.

In COBOL kan men numeriek opgemaakte velden definieren, wat vooral nuttig is voor het printen van geldsbedragen. De Basic-subroutine (r. 100-180) geeft ook bijna al deze mogelijkheden.
In $M wordt de definitie van het veld geplaatst, bijv. $M="f bb.bb9,99". De meest linkse positie is voor het teken; "9" betekent: op deze positie komt altijd een cijfer; "B" betekent: alleen zinvolle cijfers worden hier geplaatst. Speciale tekensworden alleen overgenomen als ze tussen cijfers komen testaan. Het eerste teken ordt altijd overgenomen. Het resultaat komt in $P.

```
  5REM PETER EHRLICH
 10DIMM32,P32
 20IN."GEEF MASKER"$M
 30IN."GEEF GETAL"G
 40GOS.100;P.'$P';G.20
100L=LEN(M)
120P?L=13;L=L-1;P?L=32
130IF G<0 ;P?L=CH"-"
140G=ABS(G)
150L=L-1;IF L=-1;R.
160IF M?L=CH"9"OR(G)0AND M?L=CH"B");P?L=G%10+48;G=G/10;G.150
170IF G=0 IF M?(L-1)<>CH"9";F.K=0TO L;P?K=32;N.;L=0
180P?L=M?L;G.150
```

SECTOR

Onderstaand programma leest een schijf onafhankelijk van de catalogus. Bij het opstarten wordt het tracknummer gevraagd. Aangezien een schijf 40 tracks heeft moet een getal tussen de 0 en 40 worden ingevoerd. Andere waarden worden niet geaccepteerd.
Het volgende gegeven betreft het sector nummer. Aangezien een track 10 sectoren heeft moet een getal tussen de 0 en 10 worden opgegeven. Andere getallen worden niet geaccepteerd.
Daarna komt de inhoud van de betreffende sector op het scherm. Niet ASCII waarden worden als een punt genoteerd. Een BASIC-programma wordt dan ook als een listing geprint wat de herkenbaarheid bevorderd.
Op de dan volgende vraag kan men met "J" antwoorden om de volgende sectoren ook nog te bekijken.
Indien men geen sectoren meer wil bekijken dan volgt op het antwoord "N" de vraag of men een DISAS van de betreefende sector wil zien. Hiertoe is MIRACLE van Frans van Hoesel er aan vast geplakt. Indien men geen DISAS wil hebben volgt de vraag of men de sector naar het geheugen wil hebben. Indien men met "J" antwoordt worden 57 sectoren naar het geheugen geschreven of tewel van #2900 t/m #8000. Deze 57 sectoren worden in regel 500 bepaald en kunnen gewijzigd worden naar gelang men geheugen heeft. Met behulp van dit prgramma kan men dus file's van een schijf "redden" waarvan de catalog in de vernieling is.

```
   10REM SECTOR-CONTROLE
   20REM HENK REINDERS
   30REM GRONINGEN
   50DIMCC20,DD20,LL20,TT1;F.A=0TO20;LLA=-1;CCA=-1;DDA=-1;N.
   60TT0=-1;TT1=-1;V=#70;P.$21;FORA=0TO1;P=#2300;[
start hoofd-prgrmma.
   70:DD10JSR#E223
   80LDA@#01;STA#F1     aantal sectoren op 1.
   90JSR#F7D1;]
  100?P=#0C;P=P+1
  110$P="START-TRACK";P=P+L.P;?P=#EA;P=P+1;[
  120JSRCC10;STA#EC     track-nummer.
  130CMP@#28;BCSDD10
  140:DD11JSR#F7D1;]
  150!P=#0A0D;P=P+2;$P="START-SECTOR";P=P+L.P;?P=#EA;P=P+1;[;
  160JSRCC10;STA#ED     sectoer nummer.
  161CMP@#0A;BCSDD11
  170JMPDD3
  180:DD1INC#ED         volgende sector.
  190LDA#ED;CMP@10;BNEDD3 test op maximum aantal sectoren.
  200INC#EC             volgende track.
  210LDA#EC;CMP@#28;BEQDD10 test op maximum aantal sectoren.
  220LDA@#00;STA#ED     aantal tracks op o.
  230:DD3JSR#FFED
  240JSRCC0             print een sector.
  250JSR#FFED;JSR#F7D1;]
  260$P="TRACK";P=P+L.P;?P=#EA;P=P+1;[;
  270LDA#EC;JSR#F802    print tracknummer.
  280JSR#F7D1;]
  290$P="SECTOR";P=P+L.P;?P=#EA;P=P+1;[
```

```
300LDA#ED;JSR#F802;JSR#FFED print sectornummer.
310:DD13JSR#F7D1;]
320$P="NOG EEN SECTOR ZIEN? ";P=P+L.P;?P=#EA;P=P+1;[
330:DD0JSR#FFE6;CMP@#4A;BEQDD1
340CMP@#4E;BNEDD0
350JSR#FFED;JSR#F7D1;]
360$P="DISAS ?";P=P+L.P;?P=#EA;P=P+1;[
370:DD12JSR#FFE6;CMP@#4A;BEQDD14
380CMP@#4E;BNEDD12
390BEQDD15
400:DD14JSRCC11       disas met miracle
410JMPDD13
420:DD15JSR#FFED;JSR#F7D1;]
430$P="NAAR GEHEUGEN ?";P=P+L.P;?P=#EA;P=P+1;[
440:DD5JSR#FFE6;CMP@#4A;BEQDD6
450CMP@#4E;BNEDD5
460RTS                einde hoofd-programma.
schrijf 57 sectoren naar geheugen.
470:DD6LDY@#FF;STY#C9 lage byte laadt adres.
480INY;STY#CC         minder dan 64k.
490LDA@#28;STA#CA     hoge byte laad-adres.
500LDA@#57;STA#CB     aantal sectoren.
510JSR#E75B           motor aan.
520:DD7JSR#E816;BEQDD8
530:DD9JSR#E846
540LDA@#53;JSR#E7ED   geef laadopdracht.
550JSR#E7A4;BNEDD9    controle van het resultaat.
560JSR#E839
570LDA#EC;CMP@39;BNEDD7 controle op opverschrijden aantal tracks.
580:DD8RTS            einde hoofd-programma.
schrijf de sector inhoud op scherm.
600:CC0;LDA@#E;JSR#FE55
610JSR#E75B
620:CC1JSRCC5;JSR#E226
630LDA@#00;STA#321    kolom indeling op 0.
640LDA@#28;STA#81
650LDY@#00;STY#80
660:CC2LDA(#80),Y
670INY;BEQCC3         test of alles geprint is.
680CMP@#0D;BEQCC6     test op return
690CMP@31;BCCCC4      test op kleiner dan ASCCI.
700CMP@#60;BCSCC4     test op groter dan ASCII
710JSR#FE52           schrijf ASCCI waarde.
720JMPCC2
printen van een regelnummer.
730:CC6JSR#FFED
740LDA(#80),Y;STA#25 hoge byte regelnummer naar basic-stac.
750INY;BEQCC3         test of alles geprint is.
760LDA(#80),Y;STA#16 lage byte regelnummer naar basic-stack.
770INY;BEQCC3;STY#03
780LDA@#00;STA#34;STA#27;STA#43
790JSR#C589           printen regel nummer.
800LDY#03;BEQCC3
810CPY@1;BNECC2
printen van een "." voor niet ASCCI waarde.
820:CC4LDA@#2E
830JSR#FE52
```

```
  840JMPCC2
  850:CC3RTS
sector laden op #2800.
  860:CC5JSR#E792
  870LDA@#27;STA#F7
  880LDA@#53;JSR#E7ED
  890JSR#E7A4;BNECC5
  900RTS
lezen van toesten-bord.
  910:CC10JSR#CD09
  920LDA#05;PHA
  930LDA#06;PHA;LDA#03;PHA
  940LDA@#40;STA#05
  950LDY@#01;STY#06
  960DEY;JSR#C90A
  970PLA;STA#03;PLA;STA#06
  980PLA;STA#05
  990LDA#52;RTS
Miracle een disas programma.
 1000:CC11LDA@#00;STA#70;STA#72
 1010LDY@#28;STY#71
 1020INY;STY#73
 1030LDA@#0E
 1040JSR#FE55;JSR#FFED
 1050:LL0JSRLL1;BNELL0
 1060:LL1LDY@#02
 1070:LL2LDA(#70),Y;STA#66,Y;DEY;BPLLL2
 1080:LL3INY;STY#F;LDX@#40
 1090:LL4LDA#66;CMP@#D4;BEQLL6;SEC;SBC#F1F3,Y;SEC
 1100SBC#F210,X;BNELL6;STA#65;LDY#F250,X
 1110:LL5RORA;ROR#65;DEY;BNELL5;LDY#F;AND#F1D5,Y
 1120BNELL7;LDA#F1E4,Y;AND#65;BNELL7
 1130:LL6DEX;BNELL4;CPY@#E;BNELL3;LDY@#0;LDX@#11;STY#F
 1140:LL7LDA#F202,Y;CMP@#F;BNELL8;LDA@#1
 1150:LL8;AND@#F;STA#69;STA#0;STX#6A;LDX@V;JSR#F7F1
 1160LDX@V;LDY@#0
 1170:LL9LDA(#70),Y;JSR#F7FA;JSR#FA08;BNELL10;PLA;PLA
 1180:LL10DEC#69;BPLLL9
 1190LDY#E0;JSR#F99A;LDX#6A;LDA#F194,X;STA#64;LDA#F154,X
 1200STA#65;LDY@#3
 1210:LL11LDX@#5
 1220:LL12ROL#64;ROL#65;ROLA;DEX;BNELL12;AND@#1F;CLC
 1230ADC@#3F;JSR#FFF4;DEY;BNELL11
 1240JSR#F7FD;LDY@#8;LDX#F;LDATT0,X;STA#69
 1250:LL13CPY@#6;BNELL18;TXA;BNELL15
 1260LDA#67;BPLLL14;DEX
 1270:LL14CLC;ADCV;STA#67;TXA;ADCV+1;STA#68
 1280LDX@#2;BNELL16
 1290:LL15LDX#0;BEQLL18
 1300:LL16LDA@#23;JSR#FFF4
 1310:LL17LDA#66,X;JSR#F802;DEX;BNELL17
 1320:LL18ASL#69;BCSLL19;LDATT1,Y;JSR#FFE9
 1330:LL19DEX;DEY;BPLLL13;JMP#C504;];TT(0)=P;S=P
 1340!P=#F9FFFEFF;P!4=#7F7FFFCF;P!8=#87B1CFF9;P!12=#F9B7FF
 1350P=P+15;TT(1)=P;!P=#0D;P!1=#292C5941;P!5=#40282C58;P=P+9
 1360N.;P.$6$12;@=0;P."ASSEMBLER FROM "&T" TO "&P'';@=8;E.
```

CASSETTE-OPERATING-SYSTEM

Dit verhaal probeert duidelijk te maken hoe binnen het Atom Cassette Operating System de bits uit een byte op de band komen. In een volgend verhaal zal ik dan duidelijk maken hoe ze er weer af komen.

De Atom gebruikt vanuit alle routines die iets op tape zetten de zogeheten OSBPUT. Dus bijvoorbeeld vanuit PUT, BPUT, SPUT maar ook SAVE en *SAVE.

De manual leert ons het volgende:

OSBPUT Put byte

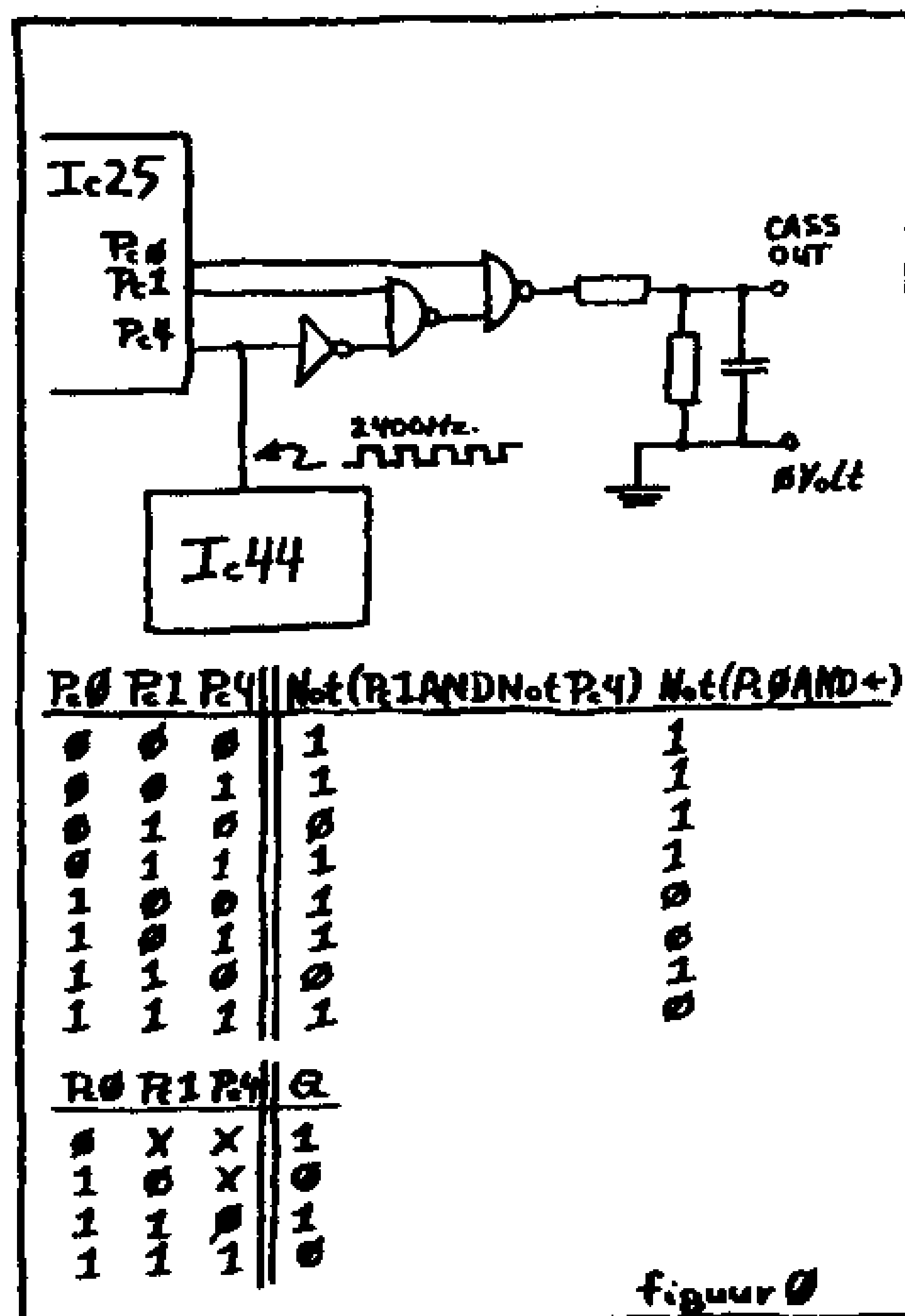
This subroutine outputs the byte in the accumulator to a sequential write file. Registers X and Y are saved. In the ATOM operating system interrupts are disabled during OSBPUT but interrupt status is restored on exit. In the Disk Operating System the file's sequential filepointer will be incremented after the byte has been saved.

In het geheugen vinden we het volgende:

```
OSBPUT
#FFD1  #6C #16 #02 JMP(#0216)

BPTVEC
#0216  #7C #FC

#FC7C  #86 #EC      STX#EC
#FC7E  #84 #C3      STY#C3
#FC80  #08          PHP
#FC81  #78          SEI
#FC82  #48          PHA
#FC83  #20 #23 #FC  JSR#FC23
#FC86  #85 #C0      STA#C0
#FC88  #20 #D8 #FC  JSR#FCD8
#FC8B  #A9 #0A      LDA@#0A LF
#FC8D  #85 #C1      STA#C1
#FC8F  #18          CLC
#FC90  #90 #0A      BCC#FC9C
#FC92  #A2 #07      LDX@#07  BEL
#FC94  #8E #02 #B0  STX#B002
#FC97  #20 #DA #FC  JSR#FCDA
#FC9A  #30 #13      BMI#FCAF
#FC9C  #A0 #04      LDY@#04  EOT
#FC9E  #A9 #04      LDA@#04  EOT
#FCA0  #8D #02 #B0  STA#B002
#FCA3  #20 #D8 #FC  JSR#FCD8
#FCA6  #EE #02 #B0  INC#B002
#FCA9  #20 #D8 #FC  JSR#FCD8
#FCAC  #88          DEY
#FCAD  #D0 #EF      BNE#FC9E
#FCAF  #38          SEC
#FCB0  #66 #C0      ROR#C0
#FCB2  #C6 #C1      DEC#C1
#FCB4  #D0 #DA      BNE#FC90
#FCB6  #A4 #C3      LDY#C3
#FCB8  #A6 #EC      LDX#EC
#FCBA  #68          PLA
#FCBB  #28          PLP
```



PØ	P1	P4	Not(P1ANDNotP4)	Not(PØAND+)
Ø	Ø	Ø	1	1
Ø	Ø	1	1	1
Ø	1	Ø	Ø	1
Ø	1	1	1	1
1	Ø	Ø	1	Ø
1	Ø	1	1	Ø
1	1	Ø	Ø	1
1	1	1	1	Ø

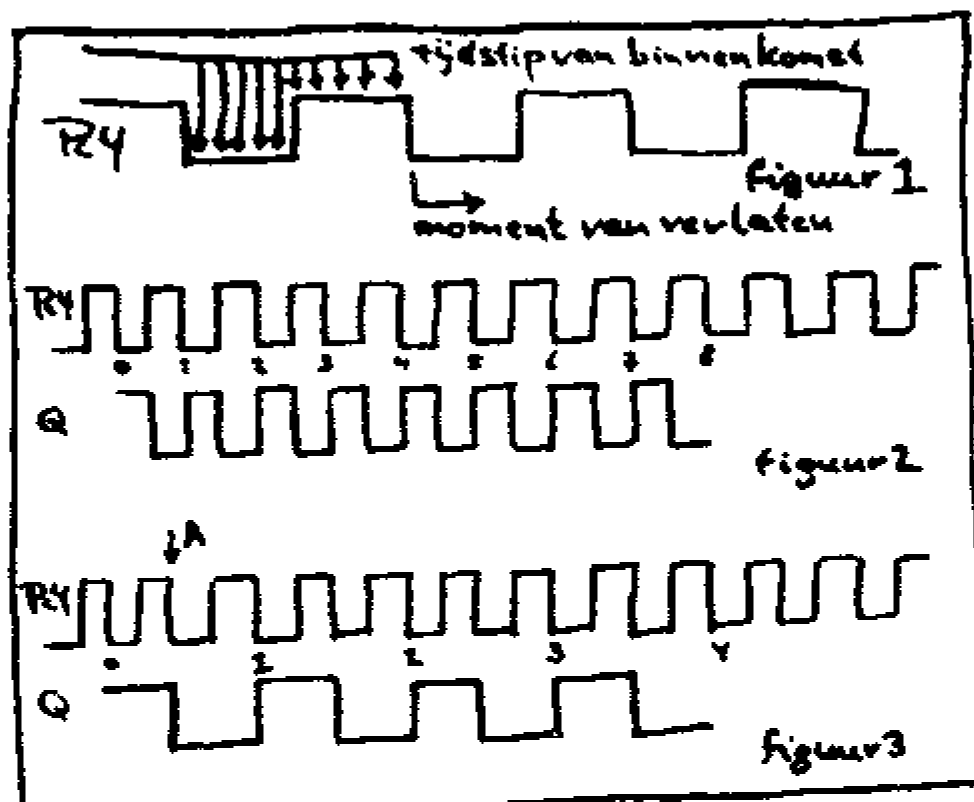
PØ	P1	P4	Q
Ø	X	X	1
1	Ø	X	Ø
1	1	Ø	1
1	1	1	Ø

figuur Ø

```
#FCBC  #60          RTS

#FCD8  #A2 #00      LDX@#00 NUL
#FCDA  #A9 #10      LDA@#10 DLE
#FCDC  #2C #02 #B0  BIT#B002
#FCDF  #F0 #FB      BEQ#FCDC
#FCE1  #2C #02 #B0  BIT#B002
#FCE4  #D0 #FB      BNE#FCE1
#FCE6  #CA          DEX
#FCE7  #10 #F3      BPL#FCDC
#FCE9  #60          RTS

#FC23  #48          PHA
#FC24  #18          CLC
#FC25  #65 #DC      ADC#DC
#FC27  #85 #DC      STA#DC
#FC29  #68          PLA
#FC2A  #60          RTS
```



De put byte routine begint op #FFD1 met een indirecte sprong via #216 naar (normaal) #FC7C. Hier worden eerst het X,Y en statusregister opgeborgen. Daarna wordt de interruptflag geset. Het gevolg hiervan is dat de 6502 een interruptrequest negeert. Dit is nodig omdat het naar de band geschreven byte verminkt zou worden als de processor halverwege het schrijven even 'weg' moet om een interrupt af te handelen. Interrupts zijn hier dus uit den boze. (Let op,Non Maskable Interrupts,die niet gedisabled kunnen worden,mogen dus niet voorkomen.)

Nu wordt de Accu op de stack opgeborgen en volgt er een subroutine aanroep. Deze subroutine,die we op #FC23 vinden,telt de Accu op bij de geheugenplaats #DC. Dit is de checksum die de Atom gebruikt om bij het teruglezen te controleren of er wel correct gelezen wordt.

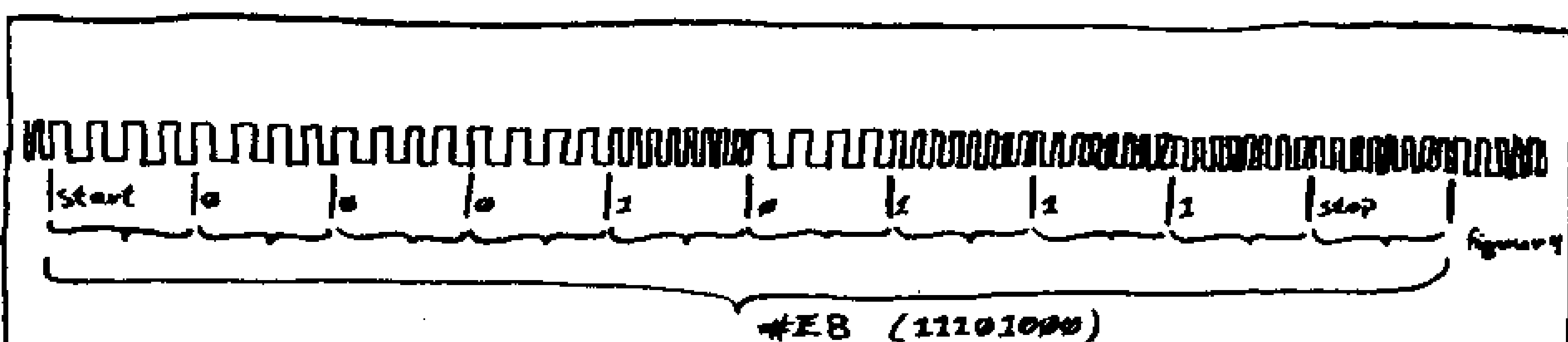
Als we terug komen van deze subroutine zien we dat de Accu opgeborgen wordt in #C0 en er wordt onmiddelijk weer een subroutine aangeroepen. Deze staat op #FCD8. Hierin wordt eerst het X register nul gemaakt en de Accu geladen met zestien. Nu volgt een 'BIT#B002'. De Zeroflag wordt dan nul als bit vier van #B002 (PC4) ook nul is. (En een als PC4 een is.) Op PC4 staat zoals uit figuur 0 blijkt een 2400 Hz. bloksolf uit IC44. De Atom wacht nu tot PC4 een is. Nu volgen twee instructies die net zo werken maar nu wordt er gewacht tot PC4 nul is. De vijf instructies van #FCDA tot en met #FCE4 zorgen er dus voor dat er op een hoog-laag overgang op PC4 gewacht wordt. (Zie figuur 1) Het X register wordt nu afgelaagd en omdat X nul was wordt er niet gesprongen en is de subroutine afgelopen.

We gaan nu verder op #FC8B. Geheugenplaats #C1 wordt nu tien gemaakt. Dit is het aantal bits dat geschreven gaat worden. Namelijk een startbit,acht databits en een stopbit. Nu volgt een 'CLC',dit omdat het startbit nul moet zijn. Op #FC90 kunnen we nu afhankelijk van de Carryflag twee kanten op. #FC9C is de routine die een nulbit op de band schrijft en #FC92 schrijft een eenbit. Het startbit dat er nu geschreven wordt geeft aan dat hier een byte begint. Waarom dat nodig is zien we straks. Zowel #FC9C als #FC92 komen na het schrijven van het bit uit op #FCAF. Hier wordt de Carry geset en #C0 wordt een maal rechtsom geroteerd. Er wordt dus een een links ingeschoven en het rechter bit komt in de Carryflag. Dit is het volgende bit dat geschreven moet worden. Nu maken we #C1 een lager en als er nog geen tien bits geschreven zijn springen we

terug naar #FC90. Doordat er telkens een een links ingeschoven wordt is het tiende geschreven bit altijd een een. Dit is het stopbit en het heeft geen vaste lengte. (Nadat de minimale lengte bereikt is blijft het gewoon staan.) Vandaar dat een startbit nodig is om het begin van een byte aan te geven. Als nu tien bits op de band geschreven zijn is OSBPUT klaar. Alle registers worden weer opgehaald en er volgt een 'RTS'.

Rest ons nog het schrijven van de bits te bekijken. Het schrijven van een eenbit vinden we op #FC92. Het X register wordt geladen met zeven en dit wordt tevens naar PC geschreven. Dat maakt PC0 en PC1 een. In figuur 0 zien we dat nu de 2400 Hz. blokgolf geinverteerd wordt doorgelaten. Er volgt nu een aanroep van de subroutine op #FCDA, dit is de eerder beschreven #FCD8 zonder dat het X register met nul wordt geladen. Het effect is dat er nu acht maal op een hoog-laag overgang op PC4 wordt gewacht. Er worden dus acht blokken doorgelaten. (figuur 2) Dit is een eenbit Het schrijven van een nulbit vinden we op #FC9C. Het Y register wordt hier vier en de Accu ook. De inhoud van de Accu gaat naar PC. PC0 en PC1 zijn dan nul en figuur 0 leert dat er een een op de uitgang staat. Nu wordt er gewacht op een hoog-laag op PC4 waarna PC wordt opgehoogd. (punt A figuur 3) Dit betekent dat PC0 een wordt. Figuur 0 laat zien dat dan de uitgang nul wordt. Daarna wordt er weer gewacht op een hoog-laag overgang. Als nu na het aflagen van het Y register geen Zeroflag geset is volgt de hele procedure opnieuw. (Totaal vier keer dus) Deze vier blokken van 1200 Hz. (tweemaal zolang als 2400 Hz.) vormen een nulbit. Als we nu een heel byte, bijvoorbeeld #E8 wegsturen dan ziet dat er uit als in figuur 4.

Voor liefhebbers van FCOS geef ik de volgende aanwijzing, waarmee het heel goed mogelijk moet zijn om de verschillen tussen de FCOS en SCOS routine te verklaren. De enen en nullen zijn hier vier maal zo kort. (immers 1200=4*300) Met andere woorden een eenbit is twee 2400 Hz. blokken en een nulbit is een 1200 Hz. blok. (figuur 5)





figuur 5

Nu zal ik proberen om duidelijk te maken hoe ze er weer af komen. Om de zaken enigszins binnen de perken te houden zal ik aannemen dat de lezer het bovengenoemde verhaal heeft gelezen.

Het gaat nu dus om OSBGET. De manual geeft ons hierover de volgende informatie:

OSBGET Get byte
The subroutine returns, in A, the next byte from a sequential read file. Registers X and Y are retained. In the ATOM operating system interrupts are disabled during OSBGET but interrupt status is restored on exit. In the Disk Operating System the file's sequential file pointer will be incremented after the byte has been read.

In het geheugen vinden we het volgende:

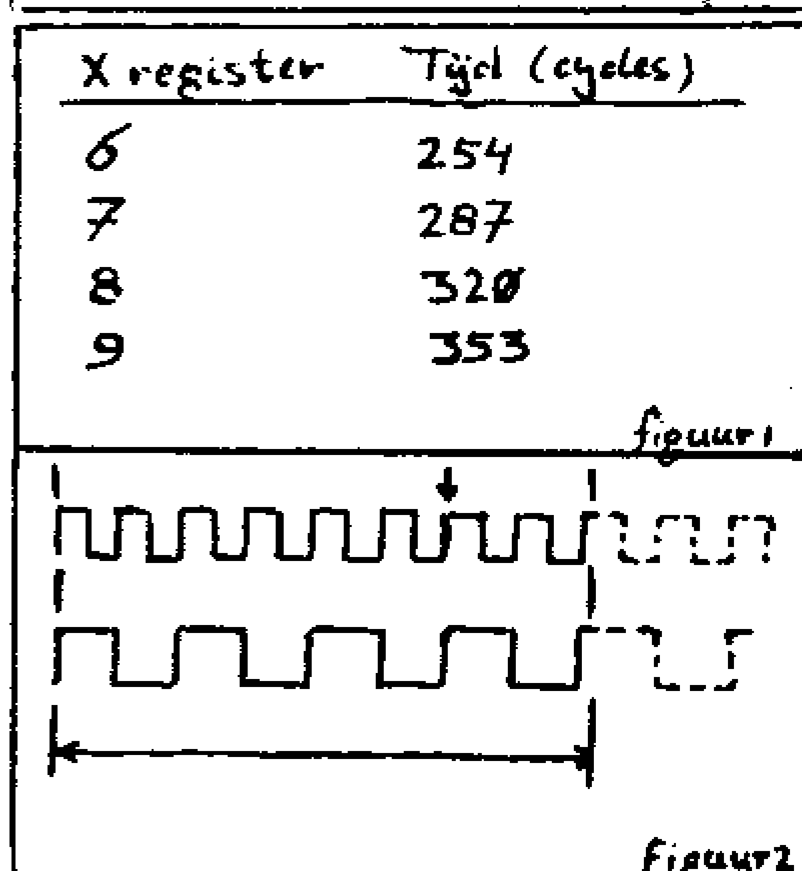
```
OSBGET
#FFD4   #6C #14 #02 JMP(#0214)

BGTVEC
#0214   #EE #FB

#FBEE   #86 #EC      STX#EC
#FBF0   #84 #C3      STY#C3
#FBF2   #08          PHP
#FBF3   #78          SEI
#FBF4   #A9 #78      LDA@#78 x
#FBF6   #85 #C0      STA#C0
#FBF8   #20 #BD #FC  JSR#FCBD
#FBFB   #90 #F7      BCC#FBF4
#FBFD   #E6 #C0      INC#C0
#FBFF   #10 #F7      BPL#FBF8
#FC01   #A9 #53      LDA@#53 S
#FC03   #85 #C4      STA#C4
#FC05   #A2 #00      LDX@#00 NUL
#FC07   #AC #02 #B0  LDY#B002
#FC0A   #20 #CD #FC  JSR#FCCD
#FC0D   #F0 #00      BEQ#FC0F
#FC0F   #F0 #01      BEQ#FC12
#FC11   #E8          INX
#FC12   #C6 #C4      DEC#C4
#FC14   #D0 #F4      BNE#FC0A
#FC16   #E0 #0C      CPX@#0C FF
#FC18   #66 #C0      ROR#C0
#FC1A   #90 #E5      BCC#FC01
#FC1C   #A5 #C0      LDA#C0
#FC1E   #28          PLP
#FC1F   #A4 #C3      LDY#C3
#FC21   #A6 #EC      LDX#EC
#FC23   #48          PHA
#FC24   #18          CLC
#FC25   #65 #DC      ADC#DC
#FC27   #85 #DC      STA#DC
#FC29   #68          PLA
#FC2A   #60          RTS

#FCBD   #A2 #00      LDX@#00 NUL
#FCBF   #AC #02 #B0  LDY#B002
#FCC2   #E8          INX
#FCC3   #F0 #07      BEQ#FCCC
```

Ic 25
8255
Pc5
cass in
SK2
0Volt
figuur 0



X register	Tijd (cycles)
6	254
7	287
8	320
9	353

figuur 1

figuur 2

```
#FCC5  #20 #CD #FC JSR#FCCD
#FCC8  #F0 #F8     BEQ#FCC2
#FCCA  #E0 #08     CPX@#08 BS
#FCCC  #60         RTS

#FCCD  #84 #C5     STY#C5
#FCCF  #AD #02 #B0 LDA#B002
#FCD2  #A8         TAY
#FCD3  #45 #C5     EOR#C5
#FCD5  #29 #20     AND@#20 SPC
#FCD7  #60         RTS
```

De get byte routine begint op #FFD4 met een indirecte sprong via #214 naar #FBEE. De eerste vier commando's heb ik al besproken (Stacker maart). Nu wordt #C0 gevuld met #78. Deze waarde lijkt vreemd maar verderop moet bit zeven van #C0 een zijn en alle andere bits nul. Als #C0 nu #78 bevat dan geeft dit na acht keer ophogen niet alleen de gewenste #80 maar ook een een in de Negativeflag. We kunnen dus zo tot acht tellen en ook nog de goede inhoud in #C0 krijgen. Nu volgt een aanroep van de subroutine op #FCBD.
Op #FCBD bevind zich een subroutine die de tijd meet tot de volgende hoog-laag of laag-hoog overgang op PC5. Dit gebeurt door het X register op te hogen tot er, met behulp van de subroutine op #FCCD, een overgang is gedetecteerd. Ik begin dus met de routine op #FCCD te bekijken.
Als we hier binnen komen bevat het Y register de 'vorige' inhoud van PC. Dit wordt opgeborgen in #C5. Nu wordt de inhoud van PC opgehaald en overgebracht naar Y. Let op, de Accu bevat nu de nieuwe inhoud van PC en Y ook (dit wordt de oude PC voor de volgende ronde). Nu wordt er een 'EXOR' uitgevoerd op de inhoud van de Accu en #C5. Het resultaat komt in de Accu. Met andere woorden de oude en de nieuwe PC worden vergeleken. Het effect is dat alle bits die veranderd zijn een worden en alle onveranderde bits nul. Als we nu een 'AND' met #20 hierop uitvoeren worden alle bits nul behalve bit 5, deze blijft onveranderd. Als er dus niets veranderd was bevat de Accu nu een nul en is de Zeroflag een. Was er wel iets verandert dan bevat de Accu #20 en is de Zeroflag nul.
Nu kijken we wat er gebeurt op #FCBD. Het X register wordt nul. Het Y register krijgt de waarde van PC en X wordt opgehoogd. Dit levert geen nul op in X dus de 'BRANCH' op #FCC3 wordt voorlopig niet uitgevoerd. Nu volgt een aanroep van de hierboven beschreven subroutine. Als PC5 onveranderd blijkt, wordt er terug gesprongen naar de 'INX' opdracht. Dit gaat door tot PC5 verandert of X 'overloopt'. Is PC5 verandert dan wordt er gekeken of X kleiner of grotergelijk acht is. In het eerste geval wordt de Carryflag nul in het tweede geval een.
Wat is nu hiervan het nut? Om dat te begrijpen gaan we de tijd berekenen die tussen #FBF8 en #FBFF verloopt als in X een bepaalde eindwaarde wordt gevonden. Dit doen we met behulp van hoofdstuk 24 uit de manual, hierin is aangegeven hoeveel cycles elk statement duurt. En omdat de Atom op een 1 MHz. clock loopt mogen we voor cycles miljoensten van seconden lezen. Ik heb een en ander voor u opgeteld en in figuur 1 gezet. Nu zien we dat als het langer als 287 cycles duurt tot de overgang op PC5 komt dan wordt X grotergelijk acht. Met andere woorden als het langer als 287E-6 sec. duurt tot de eerstvolgende overgang wordt de Carryflag een. Nu kijken we naar de 1200 Hz. en 2400 Hz. blokken die we op de band hebben gezet met OSBPUT. Een 2400 Hz. blok duurt 1/2400 sec. Dus

wachten op een overgang duurt hier maximaal 1/2*1/2400=208E-6 sec. Voor 1200 Hz. is dat 1/2*1/1200=416E-6 sec. Als er dus een eenbit (2400 Hz.) langs komt dan kan het nooit langer als 208E-6 sec. en zeker niet langer als 287E-6 sec. duren tot er een overgang komt. Voor een nulbit duurt het wel langer (416E-6)287E-6). Komt er dus een deel van een nulbit langs dan wordt de Carryflag een en anders nul.
Daarmee is de subroutine #FCBD helemaal uitgelegd en kunnen we de draad weer opvatten op #FBFB. Is de Carryflag nul (dus is het een deel van een eenbit) dan is het nog niet het gezochte startbit en begin dan maar opnieuw. Is het echter een 1200 Hz. blok tel deze dan door #C0 op te hogen. Op #FBFF wordt nu gekeken of er al acht van deze blokken zijn geweest zo niet dan moeten er nog meer blokken worden gemeten. Let op, als er een 2400 Hz. blok komt voordat er acht 1200 Hz. blokken zijn geteld wordt altijd weer helemaal opnieuw begonnen met tellen.
Nu is het startbit gelezen en bevat #C0 de waarde #80. Op #FC01 tot en met #FC16 wordt nu een databit gelezen (kom ik later op terug). Het gelezen bit zit daarna in de Carryflag en wordt op #FC18 met behulp van een 'ROR' links in #C0 ingeschoven. Rechts komt er dan een nul uit die in de Carryflag terecht komt. Nu wordt er teruggesprongen naar #FC01 om nog een databit te lezen. Dit gaat door tot er geen nul maar een een rechts uit #C0 in de Carryflag wordt geschoven. Dat gebeurt de achtste keer. Immers #C0 bevatte #80 toen we begonnen. Er zitten nu in #C0 acht databits. Dit is het te lezen byte. Op #FC1C wordt dit overgebracht naar de Accu. En klaar is OSBGET! De rest, van #FC1E tot en met #FC2A is het reeds eerder besproken herstellen der registers en bijwerken der checksum.
Rest mij het lezen van een databit te verduidelijken (Dat wordt weer rekenen). We beginnen op #FC01. #C4 Is hier een teller die om te beginnen op #53 wordt gezet. Het X register wordt 0, hierin worden de overgangen geteld die zich voordoen tijdens de #53 keer dat de lus doorlopen wordt. Het Y register krijgt de oude waarde van PC. Nu volgt de subroutine aanroep om te kijken of er een overgang is geweest. In het geval de Zeroflag hierna een is (er is dan geen overgang geweest) wordt er op #FC0D gesprongen naar #FC0F en daar nogmaals, naar #FC12. Let op, dit lijkt vreemd maar is toch zinvol. Het heeft nu zes cycles geduurt om op #FC12 te komen zonder X te veranderen. Wat gebeurt er als er wel een overgang is geweest? De Zeroflag is dan nul en er wordt tweemaal niet gesprongen waarna het X register wordt opgehoogd. Dat duurt precies ook zes cycles. Het maakt zo dus in tijd niet meer uit of er een overgang is geweest of niet. Nu zijn we op #FC12 beland, hier wordt de teller afgelaagd en als de lus nog niet #53 keer is doorlopen wordt er teruggesprongen. Is de lus vaak genoeg doorlopen dan wordt het aantal getelde overgangen vergeleken met twaalf. De Carryflag wordt een als het er twaalf of meer waren.
Dan gaan we nu de betekenis van die twaalf bekijken. Ik heb berekend (geteld) dat het hele lezen van een bit 3342E-6 sec. kost. In figuur 2 zien we deze tijdsduur uitgezet onder 1200 Hz. en 2400 Hz. blokken. (Bij de twaalfde overgang staat een pijltje.) We zien dus dat er in die tijd van de 1200 Hz. blokken nooit twaalf kunnen worden geteld (van de 2400 Hz. blokken wel). Het resultaat is dat een nulbit de Carryflag nul en een eenbit de Carryflag een maakt. En bovendien zijn we precies op tijd om het volgende bit te lezen.

SORTEREN

Naast Bubbelsort bestaan er nog andere sorteermethodes. De meeste van deze methodes zijn beter dan bubbelsort. Dat bubbelsort toch veel gebruikt wordt, komt doordat het programma relatief eenvoudig te schrijven is.

In het programma SORT (dat uiteraard gebruik maakt van P-Charme) worden verschillende sorteermethodes gedemonstreerd. De items die gesorteerd worden, staan op het scherm, zodat je kunt zien hoe het sorteren in z'n werk gaat. Sommige methodes zijn echter zo snel dat er PAUSE-statements toegevoegd moeten worden, om een en ander te kunnen volgen.

In het algemeen heten de elementen die gesorteerd moeten worden 'records' of 'items'. Men wil de elementen ordenen, meestal op grond van een bepaald veld in het record, dat dan de 'key' wordt genoemd (bv. sorteren op achternaam of op adres).
Een belangrijke vraag die men moet stellen is: waarom staan ze in de war, en waarom wil ik ze gesorteerd hebben? Zo kan men vaak voorkomen dat er gesorteerd moet worden, door bij het opslaan van de verschillende elementen er al voor te zorgen dat die in de juiste volgorde komen te staan.
Men sorteert vaak om het zoeken later gemakkelijker te maken of om geheugen beperkingen te overkomen. Dat het zoeken in een gesorteerd bestand veel sneller gaat blijkt al snel als we een naam opzoeken in het telefoonboek (dat gelukkig wel gesorteerd is). De GIRO zal waarschijnlijk sorteren om te voorkomen dat tijdens het verwerken van de giro-opdrachten het computer geheugen zo groot moet zijn dat alle rekeninghouders in de computer passen. Stel namelijk dat het giro-bestand gesorteerd op TAPE staat, dan is het voldoende om de ingekomen opdrachten (relatief klein in aantal t.o.v. het aantal rekeninghouders) ook te sorteren. Hierdoor is het mogelijk de TAPE steeds zover vooruit(!) te spoelen tot het nummer van de volgende opdracht is bereikt. Nu staan er dus op ieder moment slechts twee nummers in de computer: dat uit het giro-bestand en dat van de opdracht.

Vaak wil men dat de oorspronkelijke ordening behouden blijft, wanneer elementen identieke keys hebben. Als men bijvoorbeeld eerst een bestand op adres sorteert en dan op naam, dan wil men graag dat mensen met dezelfde naam gerangschikt blijven op adres. Voldoet een sorteermethode aan deze eis, dan zegt men dat die methode een stabiel (stable) gedrag vertoont.

Wanneer men sorteert is men continu bezig met het vergelijken (Compare) en verplaatsen (Move) van elementen.
Men tracht C en in het bijzonder M zo klein mogelijk te maken (meestal kost het verplaatsen van elementen veel meer tijd dan het vergelijken daarvan). Voor eenvoudige algoritmes (rekenmethodes) zijn C en M evenredig met het kwadraat van het aantal elementen.
Als het aantal elementen dus 10 maal zo groot wordt, worden C en M 100 maal zo groot. Voor zeer snelle algoritmes is C evenredig met N*LOG(N) en M met N, waarin N het aantal elementen is.

Enkele termen die bij sorteer-algoritmes voorkomen:

INTERNAL METHOD : elementen zitten allemaal in het werkgeheugen (Array).

EXTERNAL METHOD : elementen zitten in extern geheugen (File) bv. Disc of Tape.

IN SITU : elementen blijven in oorspronkelijk array.

NATUURLIJK GEDRAG : Als de elementen al (bijna) gesorteerd zijn, is het algoritme extra snel.

De sorteermethodes zelf zal ik hier niet bespreken. U moet dus zelf maar uitzoeken welke algoritmes snel zijn, welke een natuurlijk gedrag vertonen, of welke een stabiel gedrag hebben. Bij het bepalen van welke algoritmes snel zijn, moet men wel realiseren dat het verplaatsen van items hier waarschijnlijk vlugger gaat dan in de praktijk. Het gaat hier nl. om BYTES die verplaatst moeten worden en niet om array-elementen of strings ed. De tekens die op het scherm verschijnen die staan voor de eigenlijke items worden gebruikt als kladruimte. Deze kladruimte is bij alle algoritmes onafhankelijk van het aantal elementen.

```
 10 PROGRAM SORT
 20
 30 PROC START
 40  PRINT $7'"PRESS KEY";INKEY I
 50  CLEAR0;!8=-10;P.$30
 60  FOR I=1 TO N
 70    A?I=ABSRND%64
 80  NEXT I
 90  PRINT $30;INKEY I
100 PEND
110
120 PROC INSERTIONSORT
130  FOR I=2 TO N
140    X=A?I;A?0=X;J=I-1
150    WHILE X<A?J A?(J+1)=A?J
160      J=J-1
170    WEND
180    A?(J+1)=X
190  NEXT I
200 PEND
210
220 PROC BINARYINSERTION
230  FOR I=2 TO N
240    X=A?I;L=1;R=I-1
250    WHILE L<=R M=(L+R)/2
260      IF X<A?M R=M-1
270      IF X>=A?M L=M+1
280    WEND
290    FOR J=I-1 TO L STEP -1
300      A?(J+1)=A?J
310    NEXT J
320    A?L=X
330  NEXT I
340 PEND
350
```

```
360 PROC SELECTIONSORT
370  FOR I=1 TO N-1
380    K=I;X=A?I
390    FOR J=I+1 TO N
400      IF A?J<X K=J;X=A?J
410    NEXT J
420    A?K=A?I;A?I=X
430  NEXT I
440 PEND
450
460 PROC BUBBLESORT
470  FOR I=2 TO N
480    FOR J=N TO I STEP -1
490      IF A?(J-1)>A?J X=A?(J-1);A?(J-1)=A?J;A?J=X
500    NEXT J
510  NEXT I
520 PEND
530
540 PROC SHAKERSORT
550  L=2;R=N;K=N
560  DO
570    FOR J=R TO L STEP -1
580      IF A?(J-1)>A?J X=A?(J-1);A?(J-1)=A?J;A?J=X;K=J
590    NEXT J
600    L=K+1
610    FOR J=L TO R
620      IF A?(J-1)>A?J X=A?(J-1);A?(J-1)=A?J;A?J=X;K=J
630    NEXT J
640    R=K-1
650  UNTIL L>R
660 PEND
670
680 PROC SHELLSORT
690  T=4;H=#2800
700  H?1=15;H?2=7;H?3=3;H?4=1
710  FOR M=1 TO T
720    K=H?M;S=-K
730    FOR I=K+1 TO N
740      X=A?I;J=I-K
750      IF S=0 S=-K
760      S=S+1;A?S=X
770      WHILE X<A?J A?(J+K)=A?J
780        J=J-K
790      WEND
800      A?(J+K)=X
810    NEXT I
820  NEXT M
830 PEND
840
850 PROC SIFT
860  I=L;J=2*I;X=A?I
870g IF J>R GOTO h
880  IF J<R IF A?J<A?(J+1) J=J+1
890  IF X<A?J A?I=A?J;I=J;J=2*I;GOTO g
900h A?I=X
910 PEND
920
930 PROC HEAPSORT
```

```
 940  L=N/2+1;R=N
 950  WHILE L)1 L=L-1
 960    SIFT
 970  WEND
 980  WHILE R)1 X=A?1;A?1=A?R;A?R=X
 990    R=R-1;SIFT
1000  WEND
1010 PEND
1020
1030 PROC QUICKSORT (L,R),I,J
1040  I=L;J=R;X=A?((L+R)/2)
1050  DO
1060    WHILE A?I(X I=I+1
1070    WEND
1080    WHILEX(A?J J=J-1
1090    WEND
1100    IF I(=J W=A?I;A?I=A?J;A?J=W;I=I+1;J=J-1
1110  UNTIL I)J
1120  IF L(R QUICKSORT(L,J)
1130  IF I(R QUICKSORT(I,R)
1140 PEND
1150
1160 A=#80BF;N=#E0;PRINT $12
1170 START;PRINT"INSERTIONSORT";INSERTIONSORT
1180 START;PRINT"BINARYINSERTION";BINARYINSERTION
1190 START;PRINT"SELECTIONSORT";SELECTIONSORT
1200 START;PRINT"BUBBLESORT";BUBBLESORT
1210 START;PRINT"SHAKERSORT";SHAKERSORT
1220 START;PRINT"SHELLSORT";SHELLSORT
1230 START;PRINT"HEAPSORT";HEAPSORT
1240 START;PRINT"QUICKSORT";QUICKSORT(1,N)
1250 VTAB 15
1260 PRINT $7"  "$8$8
1270 END
```

```
 10 REM SNAPPER MET JOYSTICK
 20 DIM JJ3
 30 FOR I=0TO3;JJI=-1;NEXT I
 40 P.$21
 50 FOR I=1 TO 2;P=#3800
 60[LDA@#90;STA#B000
 70 JSR JJ0
 80 CMP@#EF;BEQ#382A
 90 CMP@#F7;BEQ#3820
100 CMP@#FB;BEQ#3825
110 CMP@#FD;BEQ#381B
120 JMP#3870
130 LDY@#01;JMP#3867
140 LDY@#02;JMP#3867
150 LDY@#03;JMP#3867
160 LDY@#00;JMP#3867
170]
180 ?#387E=#DE
190 P=#3F00
200[:JJ0
210 STX#E8;LDA#B001;STA#E9
220 LSRA;AND@#0F;TAX
230 INC#EB;LDA#EB;LSRA
240 LDA JJ2,X;BCC JJ1
250 LDA JJ3,X
260:JJ1 ASLA;ORA#E9
270 LDX#E8;RTS
280:JJ2;]
290 !P=#070F0F0F
300 P!4=#070E0F0F
310 P!8=#0B0F0B0F
320 P!12=#0F0E0D0D
330 P=P+16;[
340:JJ3;]
350 !P=#0B0F0F0F
360 P!4=#07070F0F
370 P!8=#0B0F0D0F
380 P!12=#0F0E0D0E
390 P=P+16
400 NEXT I;P.$6
410 END
```

RAM OP#0400

Bij de mensen, die een ORIGINELE ATOM DRIVE BEZITTEN, en het lage geheugen gestapeld hebben (#3C00-#3FFF) zit er 1K RAM dubbel in het systeem. Op de controller kaart zit nl. ook 1K geheugen op #3C00-#3FFF in de vorm van de IC's 20 en 21. Deze IC's worden normaal, wanneer u dit geheugengedeelte binnen de bus hebt gebracht niet gebruikt.
Dit stukje geheugen is eenvoudig op #400 tot #7FF te adresseren. Hiervoor moet de decodering op de controllerkaart veranderd worden (zie fig.1), dit gaat als volgt:

1) U verbreekt het printspoortje dat van pen 15 van IC 12 komt en naar pen 11 van IC 14 gaat (aan de soldeerzijde van de print, het dichtst bij pen 11) zie figuur 3

2) Leg nu een verbinding van het doormetaliseringsgaatje van het net onderbroken printspoor naar pin 1 van IC14 (fig. 3)

3) Onderbreek nu het printspoortje wat van pen 8 van de IC's 20 en 21 komt ,en naar pen 9 van IC 12 gaat.(aan de componentenzijde van de print vlak achter het gaatje waar het spoortje doorgemetaliseerd is.) zie figuur 2

4) Leg nu een verbinding van het doormetaliseringsgaatje van het spoortje (aan de componentenzijde vast solderen) naar pen 11 van IC 12. (aan soldeerzijde) zie figuur 2a

Het is verstandig, als u niet zo heel erg bedreven bent met de soldeerbout de IC's 12,14,20,en21 uit hun voetjes te nemen voordat u begint. Verder is het belangrijk om de onderbrekingen die u moet maken, even te controleren met een OHMmeter (moet 'oneindig' aanwijzen). Het solderen moet niet te lang duren, omdat dan de doormetaliseringen, van de gaatjes, door de warmte kapot kunnen gaan. Verder moeten de plekjes waar gesoldeerd moet worden even met een scherp mesje ontdaan worden van lak

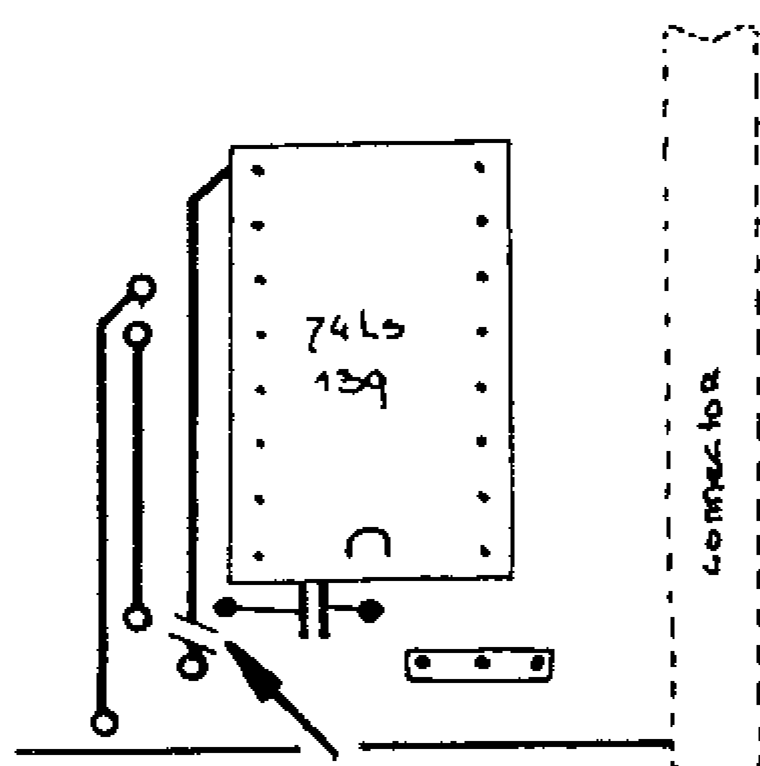


FIG 2

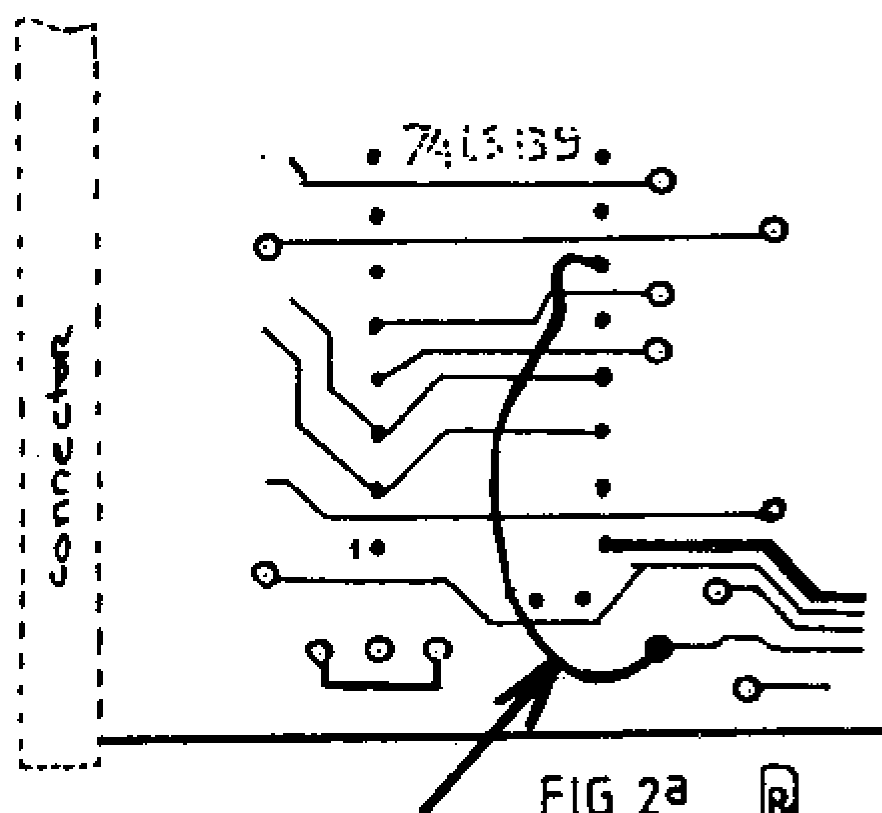


FIG 2a

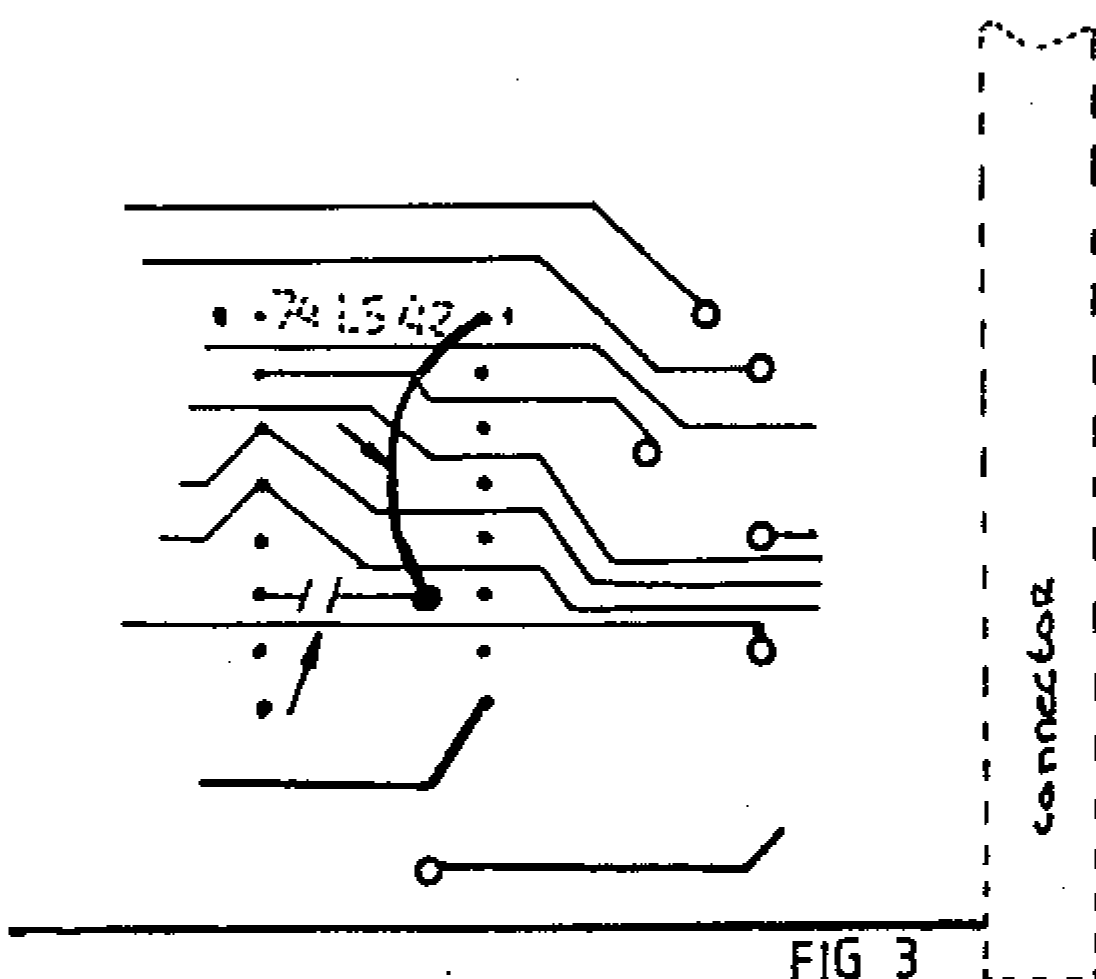


FIG 3

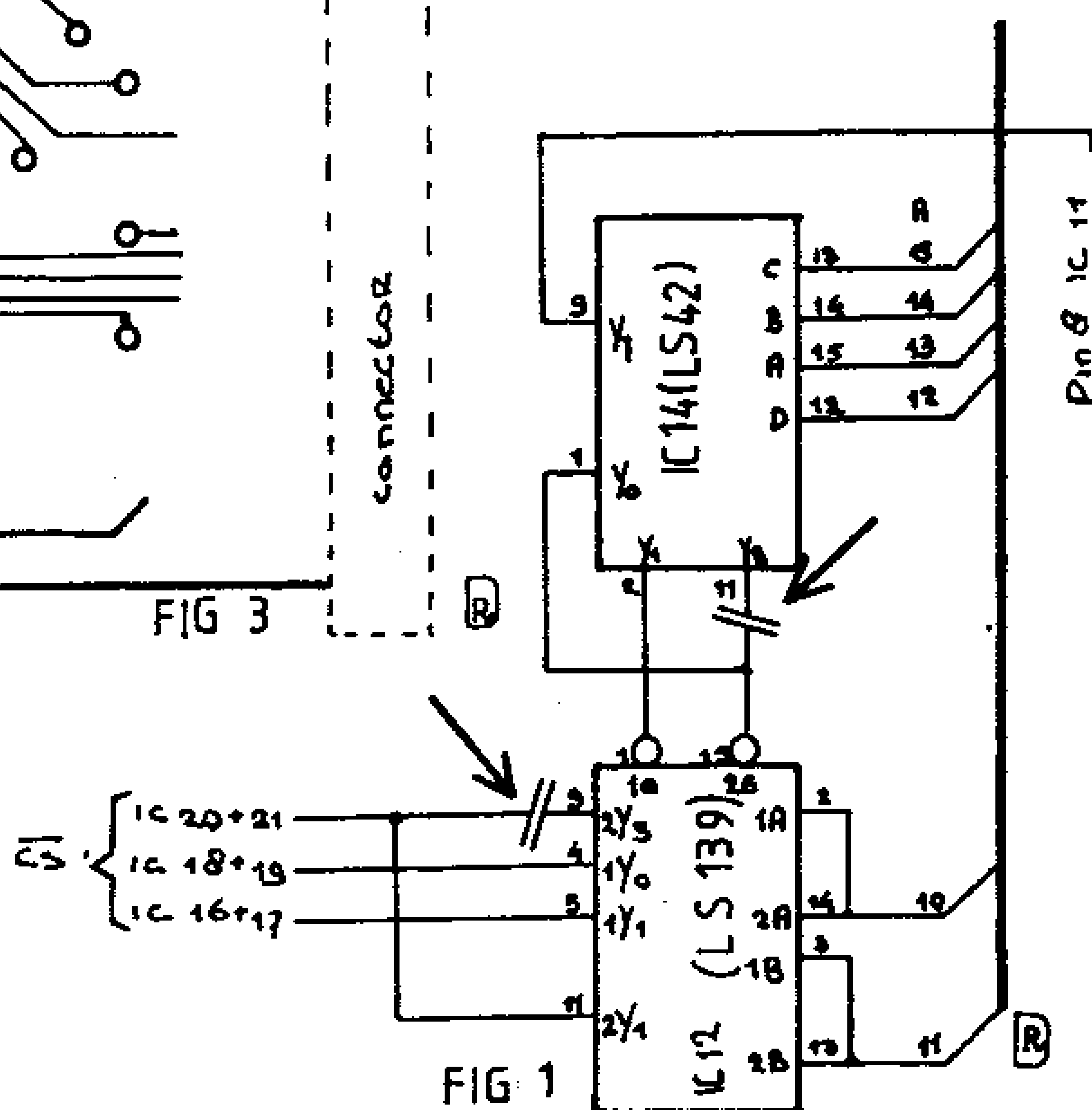


FIG 1

INTERPOLATOR

Ken je dat probleem: je wilt een grafiek tekenen, maar je kent maar een beperkt aantal punten van die grafiek.
Hoe trek je nu een mooie lijn door die bekende punten? Je zou van punt naar punt rechte stukjes lijn kunnen trekken, maar dat geeft een nogal hoekig resultaat.
Veel mooier is het een echte vloeiende lijn door de punten te trekken, maar dat wil mij nooit zo best lukken.
Vandaar het onderstaande programma "INTERPOLATOR", die deze klus in een wip klaart.

Je geeft hierbij het aantal bekende punten op en vervolgens de coordinaten van deze punten, beginnend bij het meest linkse bekende punt, daarna zijn rechter buurman, enz.enz. Als laatste worden nog de minimale en maximale waarden van X en Y opgegeven die in grafiek moeten worden gebracht en vervolgens gaat de "INTERPOLATOR" aan de slag.
Daartoe bepaalt deze de parabool die door het 1e, 2e en 3e punt loopt, vervolgens die, die door het 2e, 3e en 4e punt gaat, enz.
Wanneer nu de grafiek tussen bijv. het 6e en 7e punt wordt getekend, wordt in de buurt van het 6e punt begonnen met de parabool die hoort bij het 5e, 6e en 7e punt. In de omgeving van het 7e punt wordt de parabool aangehouden die hoort bij het 6e, 7e en 8e punt. Tussen het 6e en het 7e punt wordt een geleidelijke "overvloeier" gemaakt van de ene parabool naar de andere.

Natuurlijk zijn er nog heel veel andere interpolatie-methoden mogelijk, dus als je nog een andere, misschien fraaiere of snellere bedenkt, dan hou ik me aanbevolen.

```
 10 REM INTERPOLATOR
 20 REM AUTEUR:
 30 REM HANS HEGT
 40 REM BRABANTHOEVEN 152
 50 REM ROSMALEN
 60 IN.''"HOEVEEL PUNTEN"N;@=0
 70 IF N<3 P.'"MINIMAAL 3 PUNTEN!"';G.60
 80 FDIM%XXN,%YYN,%GG2,%AAN,%BBN,%CCN
 90 %XX0=-1E38
100 FOR T=1 TO N
110  P."X("T")="; FIN.%XXT
120 FIF%XXT<=%XX(T-1);P.'"VERKEERDE VOLGORDE!"';G.110
  4  P."Y("T")=";FIN.%YYT;P.'
140 NEXT
150 FIN."XMIN"%A;FIN."XMAX"%B;FIN."YMIN"%C;FIN."YMAX"%D
160 CLEAR4
170 FOR T=1 TO 9
180  %R=T*25.5+.5;R=%R;MOVE R,0;DRAW R,1;MOVE R,191;DRAW R,190
190  %S=T*19.1+.5;S=%S;MOVE 0,S;DRAW 1,S;MOVE 255,S;DRAW 254,S
200 NEXT T
210 %E=255/(%B-%A)
220 %G=191/(%D-%C)
230 FOR T=1 TO N
240  %X=(%XXT-%A)*%E+.5;X=%X
```

```
250  %Y=(%YYT-%C)*%G+.5;Y=%Y
260  MOVE(X-1),(Y-1);DRAW(X-1),(Y+1);DRAW(X+1),(Y+1)
270  DRAW(X+1),(Y-1);DRAW(X-1),(Y-1)
280 NEXT T
290 MOVE0,0;DRAW255,0;DRAW255,191;DRAW0,191;DRAW0,0
300 FOR K=2 TO N-1
310  %H=%XX(K-1);%I=%XX(K);%J=%XX(K+1);%L=%YY(K-1);%M=%YY(K)
320  %O=%YY(K+1);%GG0=%I-%J;%GG1=%H-%J;%GG2=%H-%I
330  %N=%GG(0)*%GG(1)*%GG(2)
340  %AA(K)=(%L*%GG(0)-%M*%GG(1)+%O*%GG(2))/%N
350  %BBK=((%M-%O)*%H*%H-(%L-%O)*%I*%I+(%L-%M)*%J*%J)/%N
360  %CCK=(%L*%GG(0)*%I*%J-%M*%GG(1)*%H*%J+%O*%GG(2)*%H*%I)/%N
370 NEXT K
380 K=1
390 FOR T=0 TO 254
400  %X=%A+T/%E
410b FIF%X>%XX(K);K=K+1;GOTOb
420  GOSUBc
430  %Y=(%Y-%C)*%G+.5;Y=%Y
440  DRAWT,Y
450 NEXT T
460 LINK#FFE3
470 END
480cIF K<3 GOTO d
490 IF K>(N-1) GOTO e
500 L=K-1;GOS.f;%R=%P;L=K;GOS.f;%S=%P
510 %Y=((%XX(K)-%X)*%R+(%X-%XX(K-1))*%S)/(%XX(K)-%XX(K-1));R.
520dL=2;GOS.f;%Y=%P;R.
530eL=N-1;GOS.f;%Y=%P;R.
540f%P=%AA(L)*%X*%X+%BB(L)*%X+%CC(L);R.
```

```
 10 REM HEXDUMP WITH ASCII
 20 REM FROM WILLOW SOFTTOOL
 30IN."START AT "I;IF I>#FFFF;E.
 40a=0;DO IF I<#1000;P.0;IF I<#100;P.0;IF I<#10;P.0
 50P.&I;FOR J=I TO I+7;P." ";IF ?J<#10;P.0
 60P.&?J;;N.;P.'"      ";FOR J=I TO I+7;C=?J
 70IF C<32;C=32
 80IF C>126;C=32
 90P."  "$C;N.;P.';I=I+8;U.I>#FFFE
100END
```

MAGISCH VIERKANT

Een magisch vierkant is een matrix gevuld met getallen, waarbij de som van de getallen op elke rij, in elke kolom en op beide diagonalen gelijk is. De getallen in het magisch vierkant moeten allemaal verschillend zijn, beginnend vanaf 1 en dan oplopend. In een magisch vierkant met zijde 3 moeten we dus de getallen 1 t/m 9 plaatsen.

Onderstaand Basic programma berekent magische vierkanten met maximum zijde 20. Een magisch vierkant met zijde 3 ziet er als volgt uit:

```
4 3 8
9 5 1
2 7 6
```

Het kleinste magisch vierkant heeft uiteraard zijde 1: 1 . Er bestaat geen magisch vierkant met zijde 2 (probeer er maar een te vinden!). De wat grotere magische vierkanten (met zijde groter dan 10) passen niet meer netjes op het beeldscherm. Als U een printer heeft, kunt U deze vierkanten het beste op papier zetten. Bezit U een 80-koloms VDU-kaart, dan kunt U de vierkanten ook netjes op het beeldscherm te voorschijn toveren. Een magisch vierkant met zijde 20 is precies 80 kolommen breed.

Het programma maakt gebruik van de P-CHARME interpreter van Frans van Hoesel, een nieuwe toolkit met o.a. procedures en functies.

```
 10 PROGRAM MAGISCH-VIERKANT
 20
 30 REM GENEREERT MAGISCHE VIERKANTEN MET ZIJDE MAX.20
 40 REM OORSPRONKELIJK GESCHREVEN IN PASCAL DOOR PETER SCHULEIN
 50 REM AANGEPAST AAN ACORN ATOM DOOR RONALD BOERS
 60 REM P-CHARME VAN F.VAN HOESEL NOODZAKELIJK
 70
 80 PROC SKIP
 90  REM DUMMY PROC VOOR LEEG ELSE-PART
100 PEND
110
120 PROC WRITE-CHAR(C,N),I
130  FOR I=1 TO N
140    PRINT $C
150  NEXT I
160 PEND
170
180 PROC WELCOME-SCREEN
190  PRINT$12
200  WRITE-CHAR(CH"*",32)
210  PRINT"*M A G I S C H  V I E R K A N T*"
220  PRINT"*";WRITE-CHAR(CH" ",30);PRINT"*"
230  PRINT"*      TYPE 0 OM TE STOPPEN     *"
240  WRITE-CHAR(CH"*",32)
250  PRINT ''
260 PEND
270
280 PROC ONEVEN(O),R,K,I
290  REM VUL ONEVEN MAGISCHE VIERKANTEN
300  R=(O+1)/2-1
```

```
310  K=O-1
320  FOR I=1 TO O*O
330    VV(R,K)=I
340    XIF I%O()0 THEN K=(K+1)%O
350        R=(R+1)%O
360    ELSE K=(K-1)%O
370  NEXT I
380 PEND
390
400 PROC VIERDEEL,K,R,H,I,J
410  FOR I=1 TO Z*Z
420    K=(I-1)%Z
430    R=(I-1)/Z
440    VV(R,K)=I
450  NEXT I ;REM VIERKANT OP VOLGORDE GEVULD
460
470  FOR R=0 TO Z-1
480    FOR K=0 TO Z/2-1
490      XIF (R%4=K%4)OR(3-(K%4)=R%4) THEN H=VV(R,K)
500          VV(R,K)=VV(Z-1-R,Z-1-K)
510          VV(Z-1-R,Z-1-K)=H
520      ELSE SKIP
530    NEXT K
540  NEXT R
550 PEND
560
570 PROC EVEN,K,H,U,W,I,J
580  REM EVEN VIERKANTEN VOLGENS STRACHEY
590  K=Z/4;H=Z/2;W=H*H
600  ONEVEN(H);REM VUL SUBVIERKANT LINKSBOVEN
610  FOR I=0 TO H-1
620    FOR J=0 TO H-1
630      VV(I+H,J+H)=VV(I,J)+W
640      VV(I,J+H)=VV(I,J)+2*W
650      VV(I+H,J)=VV(I,J)+3*W
660    NEXT J
670  NEXT I;REM LINKSBOVEN VIERKANT GECOPYEERD +SOM
680
690  FOR I=0 TO H-1
700    FOR J=0 TO K-1
710      U=VV(I,J)
720      VV(I,J)=VV(I+H,J)
730      VV(I+H,J)=U
740      U=VV(I,Z-J)
750      VV(I,Z-J)=VV(I+H,Z-J)
760      VV(I+H,Z-J)=U
770    NEXT J
780  NEXT I;REM VERWISSEL ONREGELMATIGHEDEN
790  U=VV(K,0)
800  VV(K,0)=VV(K+H,0)
810  VV(K+H,0)=U
820  U=VV(K,K)
830  VV(K,K)=VV(K+H,K)
840  VV(K+H,K)=U
850 PEND
860
870 PROC DRUK-MAGISCH-VIERKANT-AF,I,J
```

```
 880  PRINT''
 890  @=Z/11+3
 900  FOR I=0 TO Z-1
 910    FOR J=0 TO Z-1
 920      PRINT VV(I,J)
 930    NEXT J
 940    PRINT'
 950  NEXT I
 960  @=0
 970  PRINT''"MAGISCH VIERKANT MET ZIJDE:"Z'
 980 PEND
 990
1000
1010 REM HOOFDPROGRAMMA
1020 M=20;REM MAXIMALE ZIJDE MAGISCH VIERKANT
1030 DIM VV(M,M)
1040 WELCOME-SCREEN
1050 ON ERROR PRINT'"FOUTE INVOER!"'
1060 DO
1070  DO
1080    @=0
1090    PRINT'"GEEF ZIJDE MAGISCH VIERKANT:"'"(MAX."M")"
1100    INPUT Z
1110    IF Z>M THEN PRINT'"TE GROOT!"'
1120  UNTIL Z>=0 AND Z<=M
1130
1140  XIF Z>0 THEN
1150     XIF Z<>2 THEN
1160        XIF EVEN(Z) THEN
1170           XIF Z%4=0 THEN VIERDEEL
1180           ELSE EVEN
1190        ELSE ONEVEN(Z)
1200        DRUK-MAGISCH-VIERKANT-AF
1210     ELSE PRINT''"MAGISCH VIERKANT MET ZIJDE 2"'"BESTAAT NIET."'
1220  ELSE SKIP
1230  WRITE-CHAR(CH"*",32)
1240 UNTIL Z=0
1250 PRINT''"TOT ZIENS!"''
1260 END
```

FOUTJE!! FOUTJE!!

In de vorige Acorn nieuws is in het schema met verbeteringen van de kleurenkaart (pg.90) vergeten aan te geven dat condensator C4 (zie pag. 26 vorig blad) vergroot moet worden tot 470p, en R3 kortgesloten moet worden.

Rudi van Drunen

80-VDU

In Acorntjesbrood nr.1 1984 stond helemaal achteraan een sourcelisting van de systeemsoft voor de Elektuur VDU-kaart. Dit programma is gegenereerd door een zgn. sourcemaker die op enkele routines, bestemd voor een 40-kolommenkaart en aanwezig in de DOS ROM, is losgelaten met als resultaat een leesbaar en veranderbaar programma (zie ook AN 2.6 p.74-75). Het veranderen van zo hier en daar enkele parameters maakt de routine geschikt voor de VDU-kaart in naar keuze 40, 64 of 80 kolommen. Bij dit veranderen zijn er enkele foutjes gemaakt. In de regels 1440 en 1480 moet het laatste vraagteken vervangen worden door een deelstreep. Deze routine werkt (jawel!), maar heeft enkele nadelen:

1. 7 bytes zero page gebruik, terwijl we streven naar 4 bytes: #DE, #DF, #E0 en #E1, dezelfde die de Atom VDU gebruikt.
2. het programma is zo ondoorzichtig als het Noorden. Is niet verwonderlijk, maar wel hinderlijk.
3. het programma is niet in symbolische assembler geschreven. Dit is een belangrijk nadeel (hoor wie het zegt!). Ik heb geprobeerd om het programma te schaven, te poetsen en te polijsten, maar op zeker moment toch de moed opgegeven en ben opnieuw begonnen, met bijgevoegd resultaat. Alle genoemde nadelen zijn nu verdwenen.

T.a.v. punt 1 kan ik vermelden dat in #DE/#DF het startadres van het scherm staat (linkerbovenhoek), in #E0 de X-coordinaat van de cursor en in #E1 de Y-coordinaat. Volgens de datasheet van de 6845 kan het display start address, opgeslagen in de registers 12 en 13, uitgelezen worden. Dit is echter NIET het geval. Althans, niet bij de 6845P die ik heb en u waarschijnlijk ook heeft. Misschien dat het bij de 6845S wel kan, maar hoe dan ook, het programma neemt de veilige weg en definieert #DE/#DF als schaduwadres. In de routine CHAR'ADR wordt uit deze gegevens het werkelijke karakter-adres bepaald op een algemene, doch niet erg snelle wijze. Het programma gaat ervan uit dat zich in de karaktergenerator een ASCII karakterset bevindt. Er wordt dus niet met karakterwaarden gemanipuleerd, zoals dat in de Atom VDU het geval is. Volgens een eerder gelanceerd voorstel bevindt de controller zich op #BFF0/#BFF1; het geheugengebied heb ik echter op #1800- #1FFF gelocaliseerd (adreslijn A15 anders), omdat dit al op de bus zat. Ik ben daarmee mijn rommelRAM op #9800 en hoger niet kwijt en hoef niet weer aan mijn Atom te solderen. Bovendien heb ik na: ?#BFFF=0 (schakelkaart niet op WP) ;?18=#20;NEW toegang tot maar liefst 36k geheugenruimte. Van het programma valt verder nog te melden:

1. hardware scrolling gehandhaafd. Bloedje snel.
2. page mode geimplementeerd. Deze wordt na elke toetsaanslag gereset, zodat de uitvoer altijd pas na b.v. 20 regels gestopt wordt.
3. page mode kan ook afgezet worden.
4. auto repeat, functietoetsen en extra CTRL-functies kunnen eenvoudig ingebouwd worden.
5. de initiaties van de 6845 heb ik zo gekozen dat ik de monitor zo min mogelijk hoef bij te stellen als ik hem op Atom VDU of Elektuur VDU zet. Dit kan bij u anders zijn.
6. als cursor heb ik gekozen voor een matig knipperend blokje ter hoogte van een karakter. Als u hem anders wilt hebben dan zult u dat even zelf moeten aanpassen.
7. de huidige karaktergenerator bevat 64 graphics symbolen met

waarden van #00 tot #1F en #80 tot #9F. Deze symbolen kennen we van mode 0 van de Atom (2 * 3 blokjes dus).
8. er is reeds een plotroutine geschreven die m.b.v. deze symbolen een resolutie haalt van 160 (hor.) bij 72 (vert.).
9. het programma is bruikbaar in de Word Pack die daarmee zonder al te veel wijzigingen een tekst verwerkt van 23 regels bij 80 karakters. Deze tekst is met zo'n nieuwe Word Pack gemaakt en de aanschaf van een (dure) monitor was dat zeker waard. Het zou overigens zo kunnen zijn dat het gebruik van de symbolische assembler voor u problemen oplevert, omdat u het niet kunt of wilt draaien. In dat geval geef ik u de volgende punten in overweging:
1. neem deze gelegenheid te baat om de symbolische assembler te installeren. Dit kan gebeuren in de schakelkaart in eprom, inladen in RAM op #AXXX, of inladen in RAM elders in welk geval ik een speciale versie van de assembler moet maken. Het kan allemaal. De symbolische assembler is een niet te onderschatten stuk gereedschap. Gebruiken dus. U gebruikt immers ook geen schuurpapier als u een bandschuurmachine in de kast heeft liggen?
2. wilt u toch een basic assembler listing dan zult u de diverse symbolen zelf moeten veranderen m.b.v. CHANGE-STRING (Acorntjesbrood 1.4 p.23 e.v.), P-charme, de Toolbus ROM of, maar dat raad ik u niet aan, met de COPY-toets.
3. en als laatste mogelijkheid, hoewel onbevredigend voor beide partijen, kan ik het programma vertalen en u de vertaalde co- de opsturen. Voor de geinteresseerden wil ik op deze plek nog even het boek "6502 Systems programming" van Thomas G. Windeknecht noemen. Dit boek behandelt op een gedegen wijze de opbouw van de systeempro- grammatuur van een 6502-computer. Een eerste blik in dit boek roept associaties op met APL en ander abacadabra, maar een tweede blik leert ons dat er toch veel interessante zaken uit de doeken worden gedaan. De reden dat ik dit boek noem, is een sumiere beschrijving en de toepassing van de 6845 CRT-controller. De enige keer dat Thomas de plank mis slaat is op p. 93 (en ook op p. 103) waar hij stelt dat voor hardware scrolling een RAMgebied nodig is van 4 of 6k. Volgens hem zou dit dus niet mogelijk zijn op de Elektuurkaart waar immers slechts 2k RAM op zit. De bijgevoegde routine bewijst dat dit wel degelijk mogelijk is. E.e.a. heeft tot gevolg dat hij voor de scrolling routine zijn .toevlucht neemt tot het softwarematig opschuiven van het beeld- schermgeheugen. Dit is echter in de kernel-listing wel aan te passen op de plek waar SC: staat. Voor de rest is alles dermate recht-toe-recht-aan dat dat geen problemen op zal leveren.

Is het enquete formulier al terug gestuurd.

NIEUWE MICROLINE 80 KARAKTERS MET DE EDITOR-ROM!!

Het artikel van Martin Janssen in Acorn Nieuws van december 1983 over het ontwerpen van een uitgebreide karakterset voor de Microline 80 heeft mij aan het denken gezet.
Mijn "eikel" gebruik ik veelvuldig voor het maken van wiskundige teksten. Vervelend was echter dat ik voor elk wiskundig symbool voorlopig een spatie in de tekst moest opnemen om die later met de hand op te vullen. Zo ontstonden er teksten zoals:

Als A B en B C dan A C

Later werd dit dan:

Als A ⊂ B en B ⊂ C dan A ⊂ C

Ik dacht dat ik nu eindelijk van dat invullen verlost zou zijn.
Maar ..... een aantal nieuwe karakters in een eprom stoppen, is nog wel wat anders dan ze via de editor er weer uit te halen.
Wel, na flink wat denkwerk is me dat toch gelukt:
Je typt het onderstaande programma in, je brengt de editor naar #A000 onder ?#BFFF=0, je runt het programma en zet de zo gewijzigde editor eventueel weer vast in een eprom.

```
 10 REM WIJZIGINGEN WORDPACK-ROM
 20 DIMA1,LL5;F.B=0TO5;LLB=-1;N.
 30 P.$12;IN."EXTRA ROUTINES BINNEN ROM (J/N)"$A
 40 IF ?A<>#4A AND ?A<>#4E;G.30
 50 REM I.P.V. Q-ROUTINE(AAC5-AB04) MET "KORT" Q-COMMANDO
 60 IF ?A=#4A;F.B=1TO2;P=#AAC5;GOS.a;GOS.b;N.;G.c
 70 REM BUITEN ROM
 80 IN."ASSEMBLEERADRES (BV.#4000)"C
 90 F.B=1TO2;P=C;GOS.b;N.;G.c
100aP.$21
110[\"KORT" Q-COMMANDO
120       LDA@0;STA#E7;JMP#C2B6
130];R.
140bP.$21
150[\E,R,A EN B-INPUTROUTINES
160:LL0 JSR#FFE3;CMP@0;BEQLL1;JMP#FFE9
170:LL1 JSRLL2;JMP#FFF4
180:LL2 JSR#FFE3;CMP@#7F;BCSLL2;CMP@#40;BCCLL2;CLC;ADC@#60
190       RTS
200\X EN I-INPUTROUTINES
210:LL3 JSR#FFE3;CMP@0;BEQLL2
220       RTS
230\O-OUTPUTROUTINE
240:LL4 CMP@0;BMILL5;JMP#FEFB
250:LL5 INC#B002;JSR#FEFB;DEC#B002
260       RTS
270];R.
280cP.$6;IF ?A=#4E;P."EINDADRES: "&(P-1)'
290 REM WIJZIGINGEN JSR-ADRES VAN E,R,A EN B-ROUTINES
300 ?#A4B0=LL0%256;?#A4B1=LL0/256
310 ?#A4BC=LL0%256;?#A4BD=LL0/256
320 REM IDEM X EN I-ROUTINES
330 ?#A378=LL3%256;?#A379=LL3/256
340 ?#A785=LL3%256;?#A786=LL3/256
350 REM IDEM O-ROUTINE
360 ?#ACCF=LL4%256;?#ACD0=LL4/256
370 E.
```

Zoals je ziet vraagt het programma of je de extra benodigde routines binnen of buiten de editor wilt onderbrengen.

1. "Buiten de editor" heeft het voordeel dat de werking van de editor volkomen ongewijzigd blijft.
2. "Binnen de editor" tast de werking van het Q-commando aan, maar heeft het voordeel dat de hele zaak weer in een 2532 gestopt kan worden. Het Q-commando stapt dan nog wel uit de editor, maar zet geen Basic-programma meer naar #8200. Dit lijkt mij geen bezwaar omdat in Basic-programma's toch geen zelf ontworpen symbolen voorkomen. Als iemand een betere opslagplaats binnen de editor weet te vinden, kan hij natuurlijk in regel 50 een ander assembleeradres P opgeven.

Hoe nu de toetsen te bedienen opdat de nieuwe karakters uit de printer rollen?
Wel, bij alle input-commando's (dus bij E,R,A,B,X en I), werkt dat op dezelfde manier:
de eerste toets die na een CTRL@ wordt ingedrukt levert een zelf ontworpen karakter op. Op het scherm verschijnt een normaal symbool. Dat symbool is niet gelijk aan het bedoelde nieuwe karakter en ook niet aan het karakter van de ingedrukte toets. Trek je daar niets van aan; als jouw printer grafisch kan werken onder de standaardisatienorm bit C3, komt het nieuwe karakter vanzelf op papier. Dit laatste natuurlijk wel onder de voorwaarde dat je met behulp van de aanwijzigingen in het artikel van Martin Janssen die nieuwe karakters in een 2716 hebt gekregen. Dat is mij met de programmer van de club niet gelukt!!! Na het opblazen van drie 2716's kwam ik echter tot de conclusie dat een 2532 ook keurig in de printer gebruikt kan worden. De adreslijn A11 ligt in de printer toch aan 0 volt. In mijn printer zit dus nu een 2532, waarvan maar 2K gebruikt wordt. Maar toch, even er tussen door:
WIE HEEFT MET DE PROGRAMMER VAN DE CLUB WEL 2716's GEVULD EN KAN MIJ VERTELLEN HOE DAT MOET???

Even kijken of alles nu werkt:

$$x^2 + 2x^2 + y^3 \neq \alpha.\beta$$

$$\Sigma\ fi*xi = \sqrt{5}$$

$$A \cap B \notin R$$

Gelukkig, blijkbaar wel!

Tenslotte nog de vraag onder welke toetsen de nieuwe karakters nu precies zitten?
Je zult merken dat na een CTRL@ alleen de codes #40 t.e.m. #7E, dus de toetsen "@" t.e.m. "↑", geaccepteerd worden. Deze codes worden omgezet in codes die #60 groter zijn, dus in de (grafische) codes #A0 t.e.m. #DE. Deze "Japanse" codes kun je verder naar believen vervangen door je eigen karakters. Zo zit bijvoorbeeld bij mij onder de toets "a" na CTRL@ het karakter "α"; op het scherm ontstaat dan een "A".
Veel plezier met deze klus, maar verknoei er minder eproms aan dan ik!!!

Wim van der Maaten.

"Z-R-S"

Bij het programmeren in machinetaal is het vaak nodig om van een machinetaal programma te weten welke zero-page lokatie's worden gebruikt en welke vrij zijn voor eigen gebruik.
Bij een klein programmaatje is dit nog wel met de hand te doen (door disassembleren), maar bij grotere programma's is dit een hele klus, waarbij snel fouten worden gemaakt.
Onderstaand programma kan hierbij behulpzaam zijn. U geeft een geheugengebied en een zero-page gebied op. In dit geheugen gebied wordt op zoek gegaan naar intructie's die de opgegeven zeropage lokatie's gebruiken. Naar wens worden de desbetreffende instruktie's afgedrukt. Bij geindexeerde adresseringsmethode's uitkijken: in LDX @#02;STA #90,X wordt lokatie 92 gebruikt i.p.v. #90. Denk dus om de mogelijke waarden van het gebruikte index register.
Verder kunt U aan de hand van deze listing als instruktie opgevatte tabellen en teksten opsporen. Deze lijst kan op twee manieren gesorteerd zijn: op oplopend adres of op oplopende zeropage lokatie. Deze laatste mogelijkheid is de overzichtelijkste, maar neemt wel meer rekentijd in beslag. Aan het eind van deze listing volgen nog twee overzichten: het eerste bevat alle gebruikte zero-page lokatie's uit het opgegeven gebied. Als bij de adressering een geindexeerde of indirekte adresseringsmode gebruikt wordt, staat er achter de zeropage lokatie een "+" teken.
Het tweede overzicht bevat alle lokatie's uit het opgegeven gebied, die niet gebruikt worden. Hiermee kunt U gemakkelijk lokatie's uitkiezen voor gebruik in eigen programma's.

Voorbeeld U wilt weten welke zero-page lokatie's in de COS gebruikt worden:

```
** ZERO-PAGE REFERENCE SEARCH **

SEARCH FROM:?#F818 begin COS programma.
TO:?#FCE9          einde COS programma.
FOR ZERO-PAGE:?#C0 eerste zero-page van de COS.
TO:?#FF            laatste zero-page van de COS.
LISTING OF OCCURRENCES?N


ZERO-PAGE USE IN #F818 - #FCE9:
(+ = INDEXED OR INDIRECT)

 C0  C1  C2  C3  C4  C5  C9+ CA+ CB+ CC  CD  CE+ CF+
 D0  D1  D2+ D3+ D4  D5  D6+ D8+ DB  DC  DD
 E0  EA  EC  ED+ FA  FD+

NOT USED OF #C0 - #FF:
 C6  C7  C8
 D7  D9  DA  DE  DF
 E1  E2  E3  E4  E5  E6  E7  E8  E9  EB  EE  EF
 F0  F1  F2  F3  F4  F5  F6  F7  F8  F9  FB  FC  FE  FF

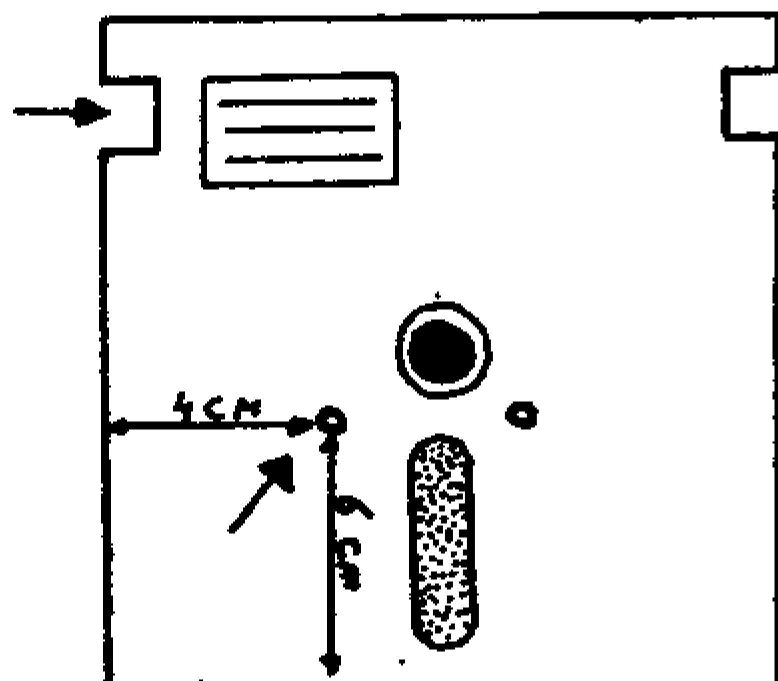
PRESS SPACE-BAR TO RESTART
```

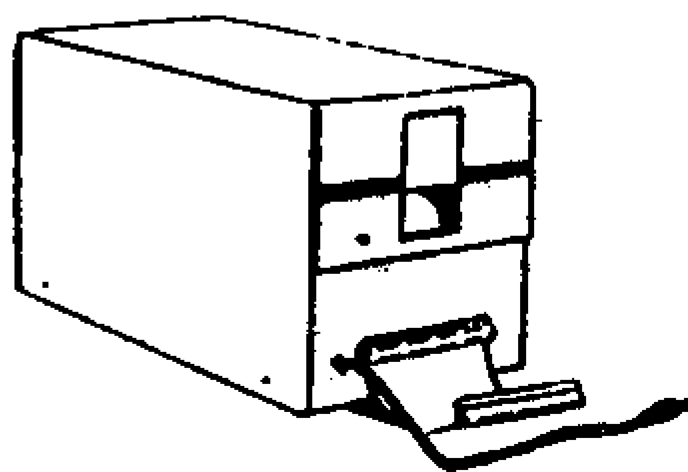
```
 10 REM /// ZERO-PAGE REFERENCE SEARCH /// V2.0
 20 O=#3800;REM BASISADRES
 30 E=#70;REM ZERO PAGE E T/M E+#F WORDEN GEBRUIKT
 40 DIM LL17,II22,TT2,T255,L5;FOR Q=0TO43;LLQ=O;N.Q
 50 P.$21;FOR I=1TO2;P=O;[
 60:LL0 LDA@0;TAX;STA T,X;INX;BNE P-4
 70:LL1 LDA E+8;STA E;LDA E+9;STA E+1
 80 INC E+4;JSR LL9
 90 LDA E+4;CMP E+5;BCC LL1;RTS
100:LL2 LDX@0
110:LL3 LDA T,X;BEQ LL4;PHA;JSR#F7FD
120 TXA;JSR#F802;PLA;JSR#FFF4
130:LL4 INX;BNE LL3;RTS
140:LL5 LDX E+10;DEX
150:LL6 INX;LDA T,X;BNE LL7;JSR#F7FD
160 TXA;JSR#F802;JSR#F7FD
170:LL7 CPX E+5;BNE LL6
180:LL8 RTS
190:LL9 JSR#C504 ESC? --) EXIT
200 JSR II0;STA E+6;LDA E+15
210 CMP@#81;BEQ LL12;CMP@#91;BEQ LL12
220 CMP@#85;BEQ LL12;CMP@#4D;BEQ LL12
230 CMP@#59;BEQ LL12;LDY E+7;CPY@2;BEQ LL17
240:LL10 LDA E;SEC;ADC E+7;STA E
250 LDA E+1;ADC@0;STA E+1;BEQ LL11
260 LDA E+2;CMP E;LDA E+3;SBC E+1;BCS LL9
270:LL11 RTS
280:LL12 LDA E+14;BNE LL13
290 LDY@1;LDA (E),Y;CMP E+4;BNE LL10;BEQ LL14
300:LL13 LDY@1;LDA (E),Y;CMP E+10;BCC LL10
310 LDA E+5;CMP (E),Y;BCC LL10
320:LL14 LDA (E),Y;TAX;LDY E+15
330 LDA@CH" ";CPY@#81;BEQ LL15
340 CPY@#82;BEQ LL15;LDA@CH"+"
350:LL15 CMP T,X;BMI LL16;STA T,X
360:LL16 LDY E+13;BNE LL10;JSR II9;JSR#FFED;JMP LL10
370:LL17 LDA (E),Y;BNE LL10;BEQ LL12
380:II0 LDX@0;LDA (E,X);TAY
390 LSRA;BCC II2;RORA;BCS II3
400 CMP@#A2;BEQ II3;AND@#87
410:II2 LSRA;TAX;LDA TT1,X;BCC II4
420 LSRA;LSRA;LSRA;LSRA
430:II4 AND@#F;BNE II5
440:II3 LDY@#80;LDA@0
450:II5 TAX;LDA TT2,X;STA E+15;AND@3;STA E+7
460 TYA;AND@#8F;TAX;TYA;LDY@3;CPX@#8A;BEQ II8
470:II6 LSRA;BCC II8;LSRA
480:II7 LSRA;ORA@#20;DEY;BNE II7;INY
490:II8 DEY;BNE II6;RTS
500:II9 LDX E;LDY E+1;JSR II20
510 JSR#F7D1;];$P=": ";P=P+2;[;LDY@0
520:II10 LDA (E),Y;JSR#F802;LDX@1
530:II11 JSR#F7FD;DEX;BNE II11
```

```
 540 CPY E+7;INY;BCC II10;LDX@3;CPY@3;BCC II11
 550 LDY E+6;LDA#F155,Y;STA E+11;LDA#F195,Y;STA E+12
 560:II12 CLD;LDA@0;LDY@5
 570:II13 ASL E+12;ROL E+11;ROLA;DEY;BNE II13
 580 ADC@#3F;JSR#FFF4;DEX;BNE II12
 590 JSR#F7FD;LDY E+7;LDX@6
 600:II14 CPX@3;BEQ II18
 610:II15 ASL E+15;BCC II16
 620 LDA TT0-1,X;JSR#FFF4
 630 LDA TT0+5,X;BEQ II16;JSR#FFF4
 640:II16 DEX;BNE II14;RTS
 650:II17 DEY;BMI II15;JSR#F802
 660:II18 LDA E+15;CMP@#E8;LDA (E),Y;BCC II17
 670 JSR II19;TAX;INX;BNE II20;INY
 680:II20 TYA;JSR#F802;TXA;JMP#F802
 690:II19 LDY E+1;TAX;BPL II21;DEY
 700:II21 ADC E;BCC II22;INY
 710:II22 RTS
 720];TT(0)=P
 730 !P=#402C292C;P!4=#00592328
 740 P!8=#00232358;P=P+12
 750 TT(1)=P
 760 !P=#30542004;P!4=#9004800D
 770 P!8=#33542203;P!12=#9004800D
 780 P!16=#33542004;P!20=#9004800D
 790 P!24=#3B542004;P!28=#9004800D
 800 P!32=#33442200;P!36=#0044C80D
 810 P!40=#33442211;P!44=#A944C80D
 820 P!48=#33442201;P!52=#9004800D
 830 P!56=#33442201;P!60=#9004800D
 840 P!64=#9A873126;P=P+68
 850 TT(2)=P
 860 !P=#82812100;P!4=#4D590000
 870 P!8=#4A869291;P!12=#9D85
 880 P=P+14
 890 N.I;P.$6
 900aP.$12"** ZERO-PAGE REFERENCE SEARCH **"''
 910 IN."SEARCH FROM:"X;E!8=X
 920 IN."TO:"Y;E!2=Y
 930 IN."FOR ZERO-PAGE:"Z;E?10=Z;E?4=Z-1
 940 IN."TO:"U;E?5=U
 950 IN."LISTING OF OCCURRENCES"$L
 960 E!12=0;IF ?L=CH"N";E!12=-1;G.l
 970 IF Z=U;G.n
 980 IN."SORTED ON ZERO-PAGE"$L
 990 E?14=(?L=CH"N");IF E?14=0;G.n
1000lE?4=E?5-1
1010nLINK LL0
1020 @=0;P.''"ZERO-PAGE USE IN #"&X" - #"&Y":"'
1030 P."(+ = INDEXED OR INDIRECT)"''
1040 LINK LL2
1050 P.''"NOT USED OF #"&Z" - #"&U":"'
1060 LINK LL5
1070 P.''"PRESS SPACE-BAR TO RESTART"
1080 LINK#FFE3;G.a
   0 (C) RONALD BOERS 24-7-1983
```

Een floppy-disk heeft twee kanten, zoals we allemaal weten. Nu rijst natuurklijk de vraag 'waarom gebruiken we er maar een??'Dit komt omdat onze drive niet geschikt is voor het dubbelzijdig gebruik van de diskette, omdat er maar een fototransistor-led paar in zit om het mbv het index-gat in de disk de electronica te synchroniseren.
Wanneer we nu de diskette omdraaien zit het index-gat niet meer op de goede plaats. Voor het gebruik van deze zijde moet er een nieuw index-gat gemaakt worden, IN DE HOES VAN DE DISKETTE, NIET IN DE SCHIJF ZELF, zonder de schijf te beschadigen.
Een manier hiervoor is: teken eerst het gat precies af (zie fig.) ,neem een stukje karton en schuif dit tussen de schijf, en het hoesje. Neem nu een zgn. HOLPIJP van 6mm (in de ijzerhandel verkrijgbaar), slijp deze event. nog iets scherper, en probeer door voorzichtig met de holpijp op hetafgetekende gat draaiend een gaatje in de hoes te ponsen.
Doe ditzelfde aan de anderezijde van de diskette,en snij met een scherp mesje ook nog een tweede write-protect sleufje eruit.
Nu kan de schijf aan de andere zijde ook gebruikt worden!!

Mij is ter ore gekomen, dat uw redacteur een tweede foto-paar in z'n TEAC drive wil zetten (in de olivetti wil dit niet zonder een gat in het chassis te boren), en daarom niet zo blij is met kopij op dubbelzijdig gemaakte diskettes.
Dit is ook op te lossen, door op het tweede index-gat een plakkertje te doen.





Onderstaand programma bepaalt van een gegeven geldbedrag alle mogelijke combinaties om dat bedrag te betalen met guldens, kwartjes, dubbeltjes en stuivers. U moet het bedrag in centen invoeren. Het programma maakt gebruik van de P-CHARME interpreter van Frans van Hoesel. Omdat de procedure BEDRAG recursief is, hangt het grootste bedrag dat U kunt laten verdelen af van de beschikbare geheugenruimte.

```
 10 PROGRAM VERDEEL-BEDRAG
 20
 30 REM DRUKT ALLE MOGELIJKE
 40 REM VERDELINGEN AF OM EEN
 50 REM BEDRAG TE BETALEN
 60 REM IN GULDENS,KWARTJES,
 70 REM DUBBELTJES EN STUIVERS
 80
 90 PROC BEDRAG(G,K,D,S)
100  @=7
110  PRINT G" "K" "D" "S'
120  N=N+1
130  X=S-2;Y=2*D+S-5;Z=5*K+2*D+S-20
140  XIF X)=0 THEN BEDRAG(G,K,D+1,X)
150  ELSE XIF Y)=0 THEN BEDRAG(G,K+1,0,Y)
160       ELSE IF Z)=0 THEN BEDRAG(G+1,0,0,Z)
170pPEND
180
190 PROC VERDEEL(B)
200  BEDRAG(0,0,0,B/5)
210 PEND
220
230 PRINT$12;N=0
240 INPUT"GEEF BEDRAG IN CENTEN:"B
250 PRINT''"    ALLE VERDELINGEN ZIJN:"''
260 PRINT"  F.1,00  F.0,25  F.0,10  F.0,05"'
270 VERDEEL(B)
280 @=0
290 PRINT''"TOTAAL "N" VERDELINGEN."''
300 END
```

DATASTAND

```
 10 REM SORTEERROUTINE VOOR DATASTAND
 20 REM BESTANDSPGM VERSIE 8D
 30 REM JOOP ENGELS,NIJMEGEN
 40 REM 3-5-1984
 50 Q=#2800;DIM KK9,MM3
 60 FOR Z=0 TO 15;KKZ=Q;NEXT
 70 B=#B0;C=B+2;D=B+4;E=B+6;T=E
 80 H=#B8;I=H+1;J=H+2;K=H+3
 90 L=H+4;S=H+5;M=H+6;V=H+7
100 P.$21;FOR Z=1 TO 2;P=Q;[
110:KK0 LDA#338;STA B;LDA#353;STA B+1;LDX@1;STX K;DEX;STX J
120:KK1 LDY J;JSR MM0;LDY K
130:MM3 STX H;STX I
140LDX@2;JSR MM0
150:KK2 LDY I;DEY
160:KK3 INY;LDA(D),Y;CMP@#20;BEQ KK3;STY I
170CMP@#61;BCC KK4;CMP@#7B;BCS KK4;EOR@#20
180:KK4 STA L;LDY H;DEY
190:KK5 INY;LDA(C),Y;CMP@#20;BEQ KK5;STY H
200CMP@#61;BCC KK6;CMP@#7B;BCS KK6;EOR@#20
210:KK6 CMP L;BCC KK8;BNE KK7
220INC H;INC I;BCS KK2
230:KK7 STA T;LDY J;LDA(B),Y;TAX;LDY K;LDA(B),Y
240LDY J;STA(B),Y;LDY K;TXA;STA(B),Y;LDA T
250:KK8 LDX@0;INC K;LDY K;CPY M;BCS KK9;CMP L;BCC MM3;BCS KK1
260:KK9 INC J;LDY J;INY;STY K;CPY M;BCC KK1;RTS
270\
280:MM0 LDA(B),Y;STA E;LDA@0;STA E+1;LDY V;LDA S;CLC
290:MM1 ADC E;BCC MM2;INC E+1;CLC
300:MM2 DEY;BNE MM1
310ASLA;ROL E+1;ASLA;ROL E+1
320ADC#301;STA E;LDA E+1;ADC#31C;STA E+1
330LDA(E),Y;STA C,X;INY;LDA(E),Y;STA C+1,X;RTS
340];P.$6;NEXT;END
```

```
 10 REM BESTANDSPGM DATASTAND
 20 REM V8D,29-2-1984
100 REM init
105 ?35=0;?36=#82
110 X=255;Y=5;V=Y+1
120 DIM LL2,S64,W(X)
130 DIM SSY,TTY,VV(X*V)
140 F.I=0TOY;DIM F16;SSI=F;N.
150 $SS0="NAAM..+..VOORL."
160 $SS1="ADRES.........."
170 $SS2="POSTCODE+PLAATS"
180 $SS3="TELEFOONNR....."
190 $SS4="GEBOORTEDATUM.."
200 $SS5="CODENUMMER....."
220 LL0=#4400;LL1=#67FF
230 LL2=20;T=LL0;M=0;GOS.d
```

```
  240 TT0=30;TT1=30;TT2=30
  250 TT3=12;TT4=6;TT5=7
  500 REM start
  510sC=#E1;D=#80;E=#86
  520 !D=#85FFE320;D!4=#6086
  530 @=0;P.$12;GOS.2000;DO
  540 P.'"WAT IS UW OPDRACHT ?"
  550jLI.D;RESTORE;U=-1
  560 DO READ $S;U=U+1
  570 IF $S="." U.1;P.$7;G.j
  580 U.?S=?E;P.$15$12
  590 GOS.(2000+1000*U);U.0
  600 DA."H","N","V","W","T"
  610 DA."B","Z","S","L","E","."
 1000 REM hulp-routines
 1010aF=0;IF M>0 R.
 1020 P.'$7"ER IS GEEN BESTAND"'"IN DE COMPUTER!"'
 1030 F=1;R.
 1040bIF M=0 R.
 1050 P.'$7"ER ZIT AL EEN BESTAND"'"IN HET GEHEUGEN !!"
 1060 P.'"WILT U DAT OVERSCHRIJVEN"'
 1070 GOS.v;IF O M=0;T=LL0
 1080 R.
 1090cP.'$S;H=S+LEN(S)+1
 1100 IN.$H;N=ABS(VAL(H))
 1110 IF N=0 OR N<=M R.
 1120 P.'"HET BESTAND KENT"'"SLECHTS "M" RECORDS..."';G.c
 1130dIF M=X R.
 1140 F.Z=M TO X-1;W?Z=Z;N.;R.
 1150vP." (J/N) ?"
 1160 DO LI.D;U.?E=74 OR ?E=78
 1170 O=0;IF ?E=74 O=1
 1180 P.$?E';R.
 1190xP.'"*** RECORD "R+1" ***";IF F P."  ("N")"
 1200 P.'';F.Z=P TO Q
 1210 P.$VV(R*V+Z)';N.;R.
 1220yF.Z=0 TO 2;B=Z+3
 1230 P.$VV(R*V+Z)
 1240 F.A=LENVV(R*V+Z) TO 31
 1250 P." ";N.;P.$VV(R*V+B)'
 1260 N.;P.';R.
 1270zF.Z=0 TO 2
 1280 P.$VV(R*V+Z)'
 1290 N.;P.''''''';R.
 1500sP.'"OP WELKE SNELHEID?"'
 1510 P.'"1.  300 BAUD"
 1520 P.'"2. 1200 BAUD"'
 1530 P.'"WAT IS UW KEUZE (1-2) ?"
 1540 DO LI.D;U.?E>48 AND ?E<51
 1550 SCOS;IF ?E=50 FCOS
 1560 P.$?E'';R.
 2000 P."acorn atom bestandsprogramma"'
 2010 P.'"U KUNT KIEZEN UIT:"'
 2020 P.'"H: HELP (TOON OPDRACHTEN)"
 2030 P.'"N: NIEUW BESTAND OF UITBREIDEN"
 2040 P.'"V: VERANDEREN IN HET BESTAND"
```

```
2050 P.'"W: RECORD WEGVEGEN IN BESTAND"
2060 P.'"T: BESTAND TONEN"
2070 P.'"B: BLADEREN IN BESTAND"
2080 P.'"Z: ZOEKEN IN BESTAND"
2090 P.'"S: SORTEREN OP SUBSTRING"
2100 P.'"L: LABELS MAKEN (PRINTER)"
2110 P.'"E: EINDE \ BESTAND WEGSCHRIJVEN"'
2120 R.
3000 P."nieuw bestand \ uitbreiden"''
3010 P."ER ZIJN 3 MOGELIJKHEDEN:"'
3020 P.'"L: BESTAND INLEZEN VAN CASSETTE"
3030 P.'"N: NIEUW BESTAND MAKEN"
3040 P.'"U: HUIDIG BESTAND UITBREIDEN"'
3050 P.'"WAT IS UW KEUZE (L,N,U) ?"
3060 DO LI.D;U.?E=76 OR ?E=78 OR ?E=85;H=?E;P.$H'
3070 IF ?E<>85 GOS.b;IF M>0 R.
3080 IF H<>76 G.n
3090 REM lees van cassette
3100 GOS.s;G=FIN"";M=BGET G
3110 F.I=0 TO M-1;W?I=BGET B
3120 F.J=0 TO Y;SGET A,T
3130 VV(I*V+J)=T;T=T+L.T+1
3140 N.;N.;R.
3150 REM nieuw\uitbreiden
3160nGOS.d;P.$12;?C=0
3170 P."U KUNT NU GEGEVENS INVOEREN."'
3180 P."GEEF NA IEDERE INVOER EEN RETURN"''
3190 P."ALS U WILT STOPPEN,"'"TYPE DAN '↑' + 'RETURN'"
3200 !#DE=#81E0;P."DRUK OP DE SPATIEBALK"
3210 DO LI.D;U.?E=32;P.$12
3220eP.''"*** RECORD "M+1" ***"''
3230 J=T;F.I=0 TO Y
3240oP.$SSI;IN.$T;L=LEN(T)
3250 IF $T="↑";T=J;R.
3260 IF T+L+1>LL1 I=Y;N.;T=J;G.f
3270 IF L>TTI P.'$7"MAXIMAAL "TTI" TEKENS!"'';G.o
3280 VV(M*V+I)=T;T=T+L+1
3290 N.;M=M+1;IF M<X G.e
3300fP.'$7"GEHEUGEN VOL!"';R.
4000 P."veranderen in het bestand"''
4010 GOS.a;IF F R.
4020 $S="GEEF HET NUMMER V/H RECORD DAT UWILT VERANDEREN "
4030 GOS.c;IF N=0 R.
4040 P.$12"U KUNT HET RECORD NU"'"REGEL NA REGEL CORRIGEREN."'
4050 P.'"BLIJFT EEN REGEL ONVERANDERD,"
4060 P.'"TYPE DAN '@' + 'RETURN'"''
4070 P."*** RECORD "N" ***"''
4080 P=N-1;F.Q=0 TO Y
4090 R=P*V+Q;H=R+1
4100rP.$VVR
4110 IF LEN(VVR)>=LL2 P.'
4120 DO P.$9;U.C.=LL2
4130 IN.$S;IF $S="@" G.m
4140 K=LEN(S);L=K-LEN(VVR)
4150 IF K>TTQ P.'$7"MAXIMAAL "TTQ" TEKENS!"'';G.r
4160 IF T+L>LL1 GOS.f;Q=Y;G.m
```

```
4170 IF N=M IF Q=Y;T=T+K+1;G.l
4180 IF L=0 G.l
4190 IF L>0 I=T-1;J=VVH;K=-1
4200 IF L<0 I=VVH;J=T-1;K=1
4210 F.Z=I TO J S.K;Z?L=?Z;N.
4220 F.Z=H TO M*V;VVZ=VVZ+L;N.
4230 T=T+L
4240l$VVR=$S
4250mN.;R.
5000 P."wegvegen"''
5010 GOS.a;IF F R.
5020 $S="WELK RECORD WILT U WEGVEGEN "
5030 GOS.c;IF N=0 R.
5040 P=0;Q=Y;R=N-1;GOS.x
5050 P.'"WILT U DIT RECORD"'"WERKELIJK WEGVEGEN"
5060 GOS.v;IF O=0 R.
5070 IF N=1 IF M=1;T=LL0;G.i
5080 P=VV(R*V);Q=VV(N*V);L=P-Q
5090 IF N=M;T=P;G.h
5100 F.Z=Q TO T-1;Z?L=?Z;N.
5110 F.Z=R*V+1 TO (M-1)*V-1
5120 VVZ=VV(Z+V)+L;N.;T=T+L
5130hI=-1;DO I=I+1;U.W?I=R
5140 F.J=0 TO M-1
5150 IF W?J>R W?J=W?J-1
5160 N.;IF I<M-1;F.J=I TO M-2;W?J=W?(J+1);N.
5170iM=M-1;R.
6000 P."tonen v/h bestand"''
6010 P=0;Q=Y
6020pGOS.a;IF F R.
6030 P.'"GESORTEERD";GOS.v;F=O
6040 P.'"U KUNT KIEZEN UIT:"'
6050 P.'"A: ALLES"
6060 P.'"E: EEN"
6070 P.'"B: BEREIK"'
6080 P.'"WAT KIEST U (A,E,B) ?";DO LI.D
6090 U.?E=65 OR ?E=66 OR ?E=69
6100 P.$?E';IF ?E=66 G.t
6110 IF ?E=65 I=1;J=M;G.u
6120 $S="WELK RECORD "
6130 GOS.c;IF N=0 R.
6140 I=N;J=N;G.u
6150t$S="EERSTE  RECORD "
6160 GOS.c;IF N=0 N=1
6170 I=N;$S="LAATSTE RECORD "
6180 GOS.c;IF N=0 N=M
6190 J=N;IF J<I P.'"KAN NIET!"';G.t
6200uP.$12;IF U=8 H=3;G.w
6210 P.'"HOE WILT U DE SUBSTRINGS ZIEN?"'
6220 P.'"1. ONDER  ELKAAR"
6230 P.'"2. ACHTER ELKAAR"
6240 P.''"WAT KIEST U (1-2) ?"
6250 DO LI.D;H=?E-48
6260 U.H>0 AND H<3;P.H'
6270wP.'"WILT U DIT PRINTEN"
6280 GOS.v;P.$12;IF O P.$2
```

```
6290 IF O=0;?#E6=1
6300 F.N=I TO J
6310 R=N-1;IF F R=W?R
6320 IF H=1 GOS.x
6330 IF H=2 GOS.y
6340 IF H=3 GOS.z
6350 N.;P.$3$15;R.
7000 P."bladeren"''
7010 GOS.a;IF F R.
7020 $S="VANAF WELK RECORD WILT U        BLADEREN "
7030 GOS.c;IF N=0 N=1
7040 ?C=0;P." ";P=0;Q=Y;R=N-1
7050 DO GOS.x;R=R+1;IF R=M R=0
7060 P.'"DRUK OP EEN TOETS, 'S' IS STOP"''';LI.D
7070 U.?E=83;?C=128;P." ";R.
8000 P."zoeken"'';P=0;Q=Y
8010 GOS.a;IF F R.
8020 IN."GEEF DE BEGINLETTER(S) "$S;IF LEN(S)=0 R.
8030 L=LEN(S)-1;F=1;?#E6=1
8040 F.R=0 TO M-1;H=R*V
8050 F.I=0 TO Y;J=H+I
8060 F.K=0 TO L
8070 IF K?S<>K?VVJ K=L;N.;G.k
8080 N.;F=0;GOS.k
8090kN.;N.;IF F P.'"TEKST NIET GEVONDEN!"'
8100 P.$15;R.
9000 P."sorteren"''
9010 IF M<2 P.'$7"ER VALT NIETS TE SORTEREN!"';R.
9020 P."U KUNT SORTEREN OP:"'
9030 F.I=0 TO Y;P.'I" "$SSI;N.
9040 P.''"WAT IS UW KEUZE (0-"Y") ?"
9050 DO LI.D;U.?E>47 AND ?E<Y+49;P.$?E'
9060 ?#BD=?E-48;?#BE=M;?#BF=V
9070 LI.#2800;R.
10000 P."labels"'';G.p
11000 P."einde"''
11010 GOS.a;IF F P.'';E.
11020 P.'"WILT U HET BESTAND WEGSCHRIJVEN"'
11030 GOS.v;IF O=0 G.11110
11040 REM op cassette opslaan
11050 GOS.s;G=FOUT""
11060 BPUT G,M;WAIT;WAIT;WAIT
11070 F.I=0 TO M-1;K=I*V
11080 BPUT B,W?I;WAIT;WAIT;WAIT
11090 F.J=0 TO Y;SPUT A,VV(K+J)
11100 WAIT;WAIT;WAIT;N.;N.
11110 P.'"WILT U WERKELIJK STOPPEN"';GOS.v;IF O=0;R.
11120 P.$12"acorn atom"''
11130 END
```

De toelichting op het bestandsprogramma DATASTAND. Het programma is onder andere geinspireerd op twee eerder verschenen bestandsprogramma's: het ene is van de hand van dhr. Groeneveld uit Leeuwarden en werd op 19-12-1982 door Hobbyskoop uitgezonden in Basicode-I. Het andere werd geschreven door Paul Smulders en gepubliceerd in de HOBBIT van jan./feb. 1984.

Uit de bewerking van deze 2 programma's resulteerde een amalgaam waarin bovendien de nodige eigen wensen en behoeftes werden ingepast, zoals specifieke afdruk-routines al naar gelang aantal en inhoud v/d substrings in een record, een sorteerroutine in machinetaal en lees/schrijf-routines van/naar cassette in 2 snelheden naar keuze. Dit alles eist nogal wat van de specifieke configuratie van de Atom:

A. een zeer groot geheugen, dat als volgt is benut:
   1. #2800-#28BF : sorteerroutine in machinetaal
   2. #2900-#4059 : het eigenlijke programma
   3. #4400-#67FF : ruimte voor de data-opslag
      -> 16K RAM-kaart nodig
   4. #8200-#9FFF : ruimte voor de array-opslag
B. de FP-ROM mag niet gebruikt worden
C. de Josbox moet aanwezig (en ingeschakeld!) zijn.

Ad A1: de sorteerroutine maakt gebruik van de ZP-adressen #B0-#BF; dit werd met name gedaan om in de ontwikkel-fase goed te kunnen testen met een disassembler/tracer; mensen die een disk hebben kunnen deze adressen echter ook verplaatsen naar bv. #70-#7F, de floating-point werkruimte. Pas dan ook regel 9060 aan.
Ad A2: hierop volgt straks een uitvoeriger toelichting.
Ad B : de FP-ROM moet wel aanwezig zijn, nl. voor het doorkoppelen van Josbox-commando's!
Ad C : de Josbox wordt gebruikt, enerzijds voor read, data, restore statements, anderzijds voor het lezen en schrijven van en naar cassette op 300 en/of 1200 Baud; beide funkties kunnen, met enkele kleine wijzigingen, ook vervuld worden door de Toolbox van Program Power.

Heeft U geen 16K-kaart, dan kan het programma eventueel toch gebruikt worden. Wijzig het daartoe bv. als volgt:
1. Kort het programma in, bv. door verwijdering van REM statements, zodat de top onder #4000 komt te liggen. Zoals U ziet wordt overigens verondersteld dat U 'varkentjes' heeft gestapeld...
2. Zet in regel 110 de variabele X (= max. aantal records) bv. op 100, waardoor bij Y=5 (= aantal substrings -1) de DIM-pointer onder adres #9000 eindigt.
3. Zet in regel 220 de variabele LL0 (= start data-opslag) op #9000 en LL1 (= eind data-opslag) op #97FF of #9FFF.
   Dan nu een toelichting bij het programma zelf!

105 : verplaats DIM-pointer naar #8200.
110 : X bevat het maximum aantal records v/h bestand. Deze waarde mag niet groter zijn dan 255! Y bevat het aantal substrings per record min 1, want substring 0 wordt ook gebruikt. De maximum waarde ligt ongeveer bij 6 of 7; bij hogere waarden moet het programma worden aangepast, met name de print-routines.
120 : LL bevat enige hulp-variabelen
      S is een input-string; W is de sorteer-array
130 : SS bevat de adressen v/d substring-namen (regel 150-200)
      TT bevat de tabulator-posities (regel 240-250)
      VV bevat de adres-pointers naar de eigenlijke data
140 : de substring-namen worden hier gedimensioneerd op max. 17 karakters.

150-200 : aantal en inhoud der substring-namen kunt U evt. naar eigen behoefte wijzigen; het programma zelf hoeft daartoe NIET veranderd te worden, uitgezonderd de afdruk-routines x, y en z.
220 : zie vorige pagina beneden, sub 3.
230 : LL2 wordt alleen gebruikt bij wijziging van een record: zie regels 4110-4120.
240-250 : TT0-TT5 bevatten de max. lengte der afzonderlijke substrings en zijn aangepast aan aantal en inhoud daarvan. Indien alle substrings op 1 regel komen te staan, mag het totaal niet meer bedragen dan 80 (max. aantal kolommen per regel op mijn printer).
510-520 : routine voor inlezen van 1 toets -> GET-funktie
590 : NB. Het programma is regel-georienteerd en mag dus absoluut NIET hernummerd worden!
1000-2000 : hulp-routines:
* a, b en c spreken voor zichzelf.
* d : zet de sorteer-array vanaf waarde M op normale, d.w.z. op ongesorteerde volgorde.
* v : hoofd-invoerroutine bij alle ja/nee vragen.
* x, y en z : afdruk-routines naar scherm en/of printer.

OPM. 1. De door mij gebruikte printer (Seikosha GP-100A) hoeft niet geinitialiseerd te worden. Heeft U bv. een Epson printer, dan is dit wel nodig. Het best kan hiervoor een subroutine geschreven worden.
2. Bij uitvoer alleen naar scherm schakelt het programma de page-mode in.

De afdruk-routines zijn specifiek voor de hier gebruikte substrings:
x : drukt record R+1 af, met de substrings P t/m Q ONDER elkaar.
y : drukt record R+1 af in 3 regels, 14telkens 2 substrings NAAST elkaar.
z : voor de vervaardiging van labels:
drukt de eerste 3 substrings van record R+1 ONDER elkaar af en slaat daarna 6 regels over.
OPM. De afstand tussen 2 labels bedraagt normaal telkens 9 regels.
s : keuze lees/schrijf-snelheid van en naar cassette.
Heeft U een PP Toolbox, verander dan 'SCOS' in 'VECTOR0' en 'FCOS' in 'VECTOR1'.
De rest van het programma spreekt (hopelijk) voor zichzelf.
Nog enige opmerkingen m.b.t. de gebruikte variabelen:
F en O : logische variabelen (0 of 1)
M : aantal in het bestand aanwezige records
T : pointer naar top data-opslag
?C : mode van de cursor
D : start-adres GET-routine
?E : ASCII-waarde invoertoets.

Apart wordt de source van het machinetaal programma gepubliceerd waarvan de objectcode op #2800 behoord te staan. Eenmaal ingevoerd kunt U vervolgens het geheel naar cassette schrijven, bv. als volgt: *SAVE"" 2800 4059 en klaar is Kees!

Nog enige slotopmerkingen:
* Gezien de sorteer-snelheid van het programma is het niet per se noodzakelijk om naast de data ook de sorteer-array naar cassette

te schrijven. Verander daartoe evt. de regels 3090-3140 en 11040-11100.

* Wanneer U het programma via de normale weg (END) of met 'escape' hebt verlaten, kunt U het ongewijzigd weer starten met 'GOTO 9'. Wilt U echter helemaal opnieuw beginnen, geef dan een 'RUN'. Tot dit laatste bent U overigens gedwongen indien U of de computer een 'reset' hebben veroorzaakt, omdat dan nl. de array-startadressen gewist zijn! Schakel tenslotte na verlaten van het programma niet de text-editor in (zo U er een hebt), want dan wordt de objectcode vanaf #2800 en wellicht zelfs het hoofdprogramma overschreven!
* Het programma werd oorspronkelijk geschreven voor het beheer van het ledenbestand van een harmonie. Dit maakte zowel de invoer van de geboortedatum als een speciaal codenummer noodzakelijk. De geboortedatum werd daartoe ingevoerd als een getal van 6 cijfers in de volgorde JJMMDD, zulks voor sorteer-doeleinden. Het codenummer telde max. 7 tekens. De betreffende waarden zijn opgeslagen in TT4 en TT5 (regel 250). Het programma is zonder meer aan te passen aan Uw eigen behoeften. Zelf heb ik gewijzigde versies gemaakt voor o.a. een bestand van LP's en cassettes en een telefoonlijst voor de studentenflat waar ik woon...

Hardware Fonds

Sinds enige tijd bestaat er een Federatief Hardware Fonds. Uit dit fonds worden de door de leden gedane ontwikkelingen van hardware bekostigd. Voordat tot aankoop van de ontwikkeling door de Federatie wordt besloten, zal eerst onderzocht worden of er bij de regio's wel belangstelling voor bestaat.
Een door de federatie benoemde hardware comissie zal daarna de ontwikkeling testen op zijn bruikbaarheid/werking.
Dit, om te voorkomen dat er met diverse kaarten in dezelfde geheugengebieden gewerkt moet worden of problemen anderszins optreden. De federatie vergoedt uiteraard alleen eenmalig de materiaalkosten etc.
Het complete ontwerp en alles wat daarbij hoort zoals bv. documentatie wordt dan eigendom van de federatie.
Voor vragen en/of opmerkingen betreffende het hardware fonds kunt U mij bellen.

A. Jongeling
secretaris F.A.C.C.
Tel: 055 - 417314

Met Hexit is het mogelijk data-tabellen te maken en te veranderen en geheugenlocaties te wijzigen.

--- STARTEN ---

Type in: HEXIT bbbb,eeee
Waarbij bbbb het beginadres is en eeee het eindadres. Als u eeee weglaat neemt Hexit 0. (dit eindadres is nodig om data tussen te voegen of weg te halen)

--- EDIT ---

Er zijn in Hexit 2 verschillende modes mogelijk nl.de EDIT-mode en de KOMMANDO-mode. Hexit begint automatisch in de EDIT mode als u bijvoorbeeld ingetoetst hebt HEXIT #2800,#28FF
ziet u het volgende:

```
2800: 48 41 4C 4C       H A L L
2804: 4F E4 2E FF       O . . .
```

enz.
Linksonder staat het woord EDIT en linksboven staat op de 4 een cursor. Aan de linkerkant staat een kolom adressen daarnaast 4 kolommen met data in HEX. Tenslotte staan aan de rechterkant 4 kolommen ASCII. Totaal staan er 14 regels met data. Met de cursor pijltjes-toetsen kunnen we de cursor over de data verplaatsen:
pijltje boven/beneden = cursor gaat naar boven.
pijltje links/rechts = cursor gaat naar rechts.
Wanneer u de SHIFT toets samen met een van de pijltjes indrukt gaat de cursor de tegenovergestelde richting uit. Merk op dat de cursor voortdurend in het hexadecimale gedeelte blijft. Als de cursor aan de rechterkant staat en u drukt op pijltje links/recht dan zal de cursor weer naar links gaan maar dan een regel lager. Als de cursor aan de boven- of onderkant van het scherm staat enu drukt op pijltje omhoog of pijltje omlaag dan zal de data respectivelijk naar beneden of naar boven scrollen. Als u een toets indrukt met een hexadecimalewaarde (0-9 of A-F) zal deze in het geheugen gezet worden en tevens op het scherm verschijnen. De cursor zal dan een HEX-karakter opschuiven. Een volgend ingetoetst HEX-karakter verschijnt dan op de volgende plaats. De twee karakters waaruit een hexadecimaal getal bestaat worden NIBBLES genoemd. Dus wanneer u 2 nibbles intoetst wordt 1 byte in het geheugen veranderd, de cursor verplaatst dan naar het volgende byte. Met de DELETE toets kun u de cursor van het ene NIBBLE naar hetandere verplaatsen.

--- ASCII ---

Als u de COPY toets indrukt verschuift de cursor naar het correponderende ASCII teken (Als dit geen normaal ASCII teken is dan zal dit op het scherm verschijnen als een punt.) Ieder ingetoetst karakter zal nu op de plaats van de cursor verschijnen en de cursor verplaatst naar het volgende byte. NB. in plaats van een NIBBEL verschuift de cursor een BYTE. De DELETE toets wordt dus gewoon als een ASCII karakter beschouwt en verschijnt op het scherm als een geinverteerd pijltje achteruit. Met de COPY toets kan de cursor weer naar het HEX-deel verplaatst worden.

--- KOMMANDO ---

Wanneer u in de EDIT-mode de ESCAPE indrukt komt u in de KOMMANDO-mode te staan.
Onderaan het scherm verschijnt nu de regel:
KOMMANDO (B/I/E/S/G/ESC/Z/DEL/):U kunt nu een van deze letters intoetsen.

*** Kommando's ***

B.....terug naar BASIC

Onderaan het scherm verschijnt:
BASIC:WEET JE HET ZEKER (J/N)
U moet dan Ja of Nee intoetsen.
I.....Invoegen van karakters
De data wordt vanaf de geheugenplaats tot aan het eindadres opgeschoven en op de plaats waar de cursor staat komt het getal 00.
Als het eindadres kleiner of gelijk is aan het beginadres zal de computer eerst vragen EINDADRES:
U moet dan eerst een eindadres invoeren,daarna gaat HEHIT weer naar de EDIT mode.
Als het eindadres groter is dan het beginadres komt onderaan het scherm te staan:
EINDADRES:#eeee (J/N):
Wanneer u J indrukt zal HEXIT een karakter invoegen.
Als u N indrukt vraagt HEXIT een nieuw eindadres.
Wanneer u J indrukt zal de vraag nog een keer gesteld worden.U kunt zoveel plaatsen vrijmaken als u wilt (eventueel met REPT toets).

E.....Eindadres invoeren

Hexit vraagt alleen:
EINDADRES:
Nadat u een eindadres in HEX hebt ingevoerd gaat HEXIT weer naar de EDIT mode.

S.....Start tekst

Hexit gaat weer terug naar het adres wat u met G ingevoerd hebt of naar het adres bbbb.

G.....Ga naar

De computer vraagt:
GA NAAR:
Je moet dan een (hexadecimaal) adres invoeren,HEXIT gaat dan weer naar EDIT mode en begint op het zojuist ingevoerde adres.

**** BELANGRIJK ****

Als HEXIT om een adres vraagt (GA NAAR of EINDADRES) dan zijn er nog 2 mogelijkheden nl.
RETURN toets:Adres wordt gelijk aan 0.
ESCAPE toets:Adres wordt niet veranderd.

Z.....Einde data

Hexit zorgt dat het eindadres rechtsonder komt te staan en de cursor linksonder.De cursor blijft in het HEX-deel of in de ASCII-deel!

ESC...Edit-mode

Keer weer terug naar de EDIT-mode.

DEL...Wissen van een BYTE

Het karakter dat onder de cursor wordt weggehaald en de rest van de data tot aan het eindadres wordt aangeschoven.
De computer stelt dezelfde vragen als bij 'I'.

Er is een kommando dat niet in de lijst staat en dat is het vraagteken:?
Wanneer u dit intoetst verschijnt op het scherm een beknopte verklaring van de mogelijke kommando's

ZORG ERVOOR DAT U ALLEEN KOMMANDO'S INTOETST ALS U IN DE KOMMANDO MODE ZIT.ANDERS KUNNEN DE INGETOETSTE KARAKTERS IN HET GEHEUGEN GEZET WORDEN EN ZO DE DATA VERNIETIGEN. LET OOK OP DAT U GEEN HEXADECIMALE GETALLEN INTOETST ALS DE CURSOR IN HET ASCII GEDEELTE STAAT OF ANDERSOM.

```
   5 REM LADEN IN #8200
  10 P.$12;?#E1=0;P."  "
  20 P."edit"'
  30P."CURSOR TOETSEN :BEWEEG CURSOR"'
  40P."COPY    :ASCCII/HEXADECIMAAL"'
  50P."DELETE :LINKER/RECHTER HEX KAR."'
  60P."ESC    :KOMMANDO"'
  70P."kommando"'
  80P."B       :BASIC"'
  90P."I       :INVOEGEN (TOT EINDADRES)"
 100P."DELETE :WISSEN (TOT EINDADRES)"'
 110P."E       :EINDADRES VERANDEREN"'
 120P."G       :GA NAAR"'
 130P."ESC    :EDIT"'
 140P."S       :GA NAAR BEGINADRES"'
 150P."Z       :GA NAAR EINDADRES"'
 160P."VOOR HERHALING I/DEL:DRUK OP J"'
 170P."DRUK OP DE SPATIEBALK"
 180 F.X=#81E0 TO #81FF;?X=?X:#80;N.
 190COPY#8000,#81FF,S
 200LI.#FE94;P.$12
 210LI.VV20;#2000;E.

   0 REM AUTEUR ONBEKEND
   5 REM LADEN IN #2900
  10 *LO."DHEX"
  20 ?#24=#96;?#23=0
  30 @=0;ON.G.60
  40 P."DRUK RETURN VOOR:"&A'
  50 IN."BEGIN ADRES"A
  60 P.$12$21;T=#2800
  70 DIMVV85;F.X=0TO85;VVX=#FFFF;N.;F.X=0TO1;P=A
  80 !T=#110E0B08;T!4=#1F1D1B19
  90 S=T+8
 100[:VV0
 110 LDA@14;STA#88
 120 LDA#80;STA#86;LDA#81;STA#87
 130:VV1 LDA@#80;STA#85
 140 LDX@0;STX#84
 150:VV2 LDA@#3A;STA(#84,X);LDY@0
 160 BIT#B002;BMIP-3
 170 INC#84;JSR VV9 ADRES
 180 INC#84;INC#84
 190:VV3 INC#84
 200 LDA(#86),Y
 210 JSR VV10
 220 INY;CPY@4;BNEVV3
 230 LDA@3;ADC#84;STA#84
 240 LDY@0
 250:VV4 INC#84;INC#84
 260 LDA(#86),Y
 270 BMI VV5
 280 CMP@#20;BCS VV6
 290:VV5 LDA@#2E
 300:VV6 CLC;ADC@#20;BMI VV7;EOR@#60
 310:VV7 STA(#84,X);INY;CPY@4;BNE VV4
 320 CLC;TYA;ADC#86;STA#86;BCC VV8;INC#87
```

```
330:VV8 INC#84;BNEP+4;INC#85
340 DEC#88;BNEVV2;RTS
350:VV9 LDA#87;JSR VV10;LDA#86
360:VV10 PHA;LSRA;LSRA;LSRA;LSRA
370 JSRVV11;PLA
380:VV11 AND@#F;CMP@#A;BCC VV12
390 SBC@9;BNE VV13
400:VV12 ADC@#30
410:VV13 STA(#84,X)
420 INC#84;RTS
430:VV14 PHP;CLD;STX#E4;STY#E5
440:VV15 BIT#B002;BVC VV16
450 JSR#FE71;BCC VV15
460:VV16 JSR#FB8A
470:VV17 JSR#FE71;BCS VV17
480 JSR#FE71;BCS VV17
490 TYA;CMP@6;BEQ VV18
500     CMP@7;BEQ VV18
510     CMP@#E;BEQ VV19
520 JMP#FEB2
530:VV18 AND@1;ROL#B001;ROL A
540 JMP#FE60
550:VV19 LDA@4;JMP#FE60
560:VV29 SEC;LDA#89;SBC#80;STA#72;AND@#3C;LDX@#10;STX#71
570 ASLA;ROL#71;ASLA;ROL#71;ASLA;ROL#71;STA#70;LDA#72
580 AND@3;CLC;ADC#7F
590 AND@#7;TAX;CLC;LDAT,X;ADC#70;STA#70;LDX@0;BIT#7F;BPL P+4
600 INC#70;LDA(#70,X);EOR@#80;STA(#70,X);RTS
610:VV20 LDA@#52;STA#208;LDA@#FE;STA#209
620 LDA@0;STA#82;STA#83
630 JSR#C8BC;JSR#C231;BCS VV83;JSR#C8BC
640 LDY@#82;JSR#C3CD
650:VV83 JSR#C3CB;LDA#52;STA#80;STA#89;LDA#53;STA#81;STA#8A
660 LDA@0;STA#7F
670:VV85 LDA@12;JSR#FFF4
680 LDX@15;LDA@10;JSR#FFF4;DEX;BNEP-4;JSRVV0;JSRVV29;JMPVV62
690:VV21 JSRVV14
700 CMP@#1B;BNEP+5;JMP VV63
710 TAX;BEQ VV25;DEX;BEQ VV30
720 DEX;BEQ VV31 \ BENEDEN
730 DEX;BNE P+5;JMPVV35
740 DEX;BEQ P+5;JMPVV39
750 JSRVV29;LDA#7F;AND@4;EOR@4;STA#7F;JSRVV29;JMPVV21
760:VV30 JSRVV29
770 LDA#72;CMP@#37;BNEVV23
780 CLC;LDA#80;ADC@4;STA#80
790 BCC VV22;INC#81
800:VV22 JSRVV0
810:VV23 INC#89;BNE VV24;INC#8A
820:VV24 JSR VV29;JMP VV21
830:VV25 JSR VV29
840 LDA#72;BNE VV27
850 SEC;LDA#80;SBC@4;STA#80;BCS VV26;DEC#81
860:VV26 JSRVV0
870:VV27 LDA#89;BNE VV28
880 DEC#8A
890:VV28 DEC#89;JSRVV29;JMP VV21
```

```
 900:VV31 JSRVV29;LDA#72;CMP@#34;BCC VV33
 910 CLC;LDA#80;ADC@4;STA#80;BCC VV32;INC#81
 920:VV32 JSRVV0
 930:VV33 CLC;LDA@4;ADC#89;STA#89;BCC VV34;INC#8A
 940:VV34 JSRVV29;JMP VV21
 950:VV35 JSRVV29;LDA#72;CMP@4;BCS VV37
 960 SEC;LDA#80;SBC@4;STA#80;BCS VV36;DEC#81
 970:VV36 JSRVV0
 980:VV37 SEC;LDA#89;SBC@4;STA#89;BCS VV38;DEC#8A
 990:VV38 JSRVV29;JMPVV21
1000:VV39 LDX#7F;CPX@4;BNE VV40
1010 LDX@0;STA(#89,X)
1020 JSRVV0;JSRVV29;JMPVV30
1030:VV40 CMP@#7F;BEQ VV43
1040 JSR#F87E;BCC VV41;LDA@7;JSR#FFF4;JMPVV21
1050:VV41 LDX@0;BIT#7F;BPL VV42;STA#88
1060 LDA(#89,X)
1070 AND@#F0;ORA#88;STA(#89,X)
1080 JSRVV0;LDA#7F;EOR@#80;STA#7F
1090 JSRVV29;JMPVV30
1100:VV42 ASLA;ASLA;ASLA;ASLA;STA#88;LDA(#89,X)
1110 AND@#F;ORA#88;STA(#89,X)
1120:VV43 JSRVV0;LDA#7F;EOR@#80;STA#7F
1130 JSRVV29;JMPVV21
1140:VV52 JMPVV62
1150:VV50 \ input buffer
1160 LDA@#3A;JSR#FFF4
1170 LDY@0;STY#64;STY#65
1180:VV51 JSRVV14;TAX
1190 CMP@#1B;BEQ VV52
1200 CMP@#D;BEQ VV57
1210 CMP@#7F;BNE VV53
1220 CPY@0;BEQ VV58
1230:VV55 JSR#FFF4;DEY;BPL VV51
1240:VV53 JSR#F87E;BCC VV54
1250:VV58 LDA@7;JSR#FFF4;JMPVV51
1260:VV54 STA#60,Y;TXA;JSR#FFF4;INY;CPY@4;BNE VV51
1270:VV56 JSRVV14;CMP@#7F;BEQVV55;CMP@13;BNE VV56
1280:VV57 LDA@#FF;STA#60,Y;LDY@0
1290:VV60 LDX@3;LDA#60,Y
1300 BMI VV61-1
1310 ASLA;ASLA;ASLA;ASLA
1320:VV59 ASLA;ROL#64;ROL#65
1330 DEX;BPL VV59;INY;CPY@4;BNEVV60
1340 RTS
1350:VV61 LDA@13;JSR#FFF4 clear
1360 LDX@#E0;LDA@#20;STA#8100,X;INX;BNEP-4;RTS
1370:VV62 \ edit mode
1380 JSRVV61;JSR#F7D1;]$P="EDIT";P=P+L.P;[LDA@#20;STA#81E4
1390LDX@#FF;TXS;JMPVV21
1400:VV63JSRVV61;JSR#F7D1;]$P="KOMMANDO (B/I/E/S/G/ESC/Z/DEL):"
1410P=P+L.P;[NOP
1420:VV64 JSRVV14;CMP@#1B;BEQ VV62;CMP@#42;BNE VV67
1430 JSRVV61;JSR#F7D1;]$P="BASIC:WEET JE HET ZEKER (J/N)"
1440P=P+L.P;[NOP
1450:VV66 JSR VV14;CMP@#4A;BNE VV65;JSR#F7D1;];?P=12;P=P+1
1460 $P="  BYE FROM HEXIT  ";P=P+L.P+3;P?-1=10;P?-2=10;[NOP
```

```
1470 JMP#C2CF
1480:VV65 CMP@#4E;BNE VV66
1490 JMPVV62
1500:VV67 CMP@#47;BNE VV68
1510:VV72
1520 JSRVV61;JSR#F7D1;]$P="GA NAAR";P=P+L.P;[NOP;JSRVV50
1530 LDA#64;STA#80;STA#89;STA#8B;LDA#65;STA#81;STA#8A
1540 STA#8C;JSRVV0;JSRVV29
1550 JMPVV69
1560:VV68 CMP@CH"E";BNE VV70
1570:VV69JSRVV61;JSR#F7D1;]$P="EIND ADRES";P=P+L.P;[NOP;JSRVV50
1580 LDA#64;STA#82;LDA#65;STA#83;JMP VV62
1590:VV70 CMP@CH"Z";BNEVV71
1600 SEC
1610 LDA#82;SBC@3;STA#89;LDA#83;STA#8A;BCS P+4;DEC#8A
1620 SEC;LDA#82;SBC@#37;STA#80;LDA#83;STA#81;BCS P+4;DEC#81
1630 JSRVV0;JSRVV29;JMPVV62
1640:VV71 CMP@CH"S";BNEVV78
1650 JSRVV29;LDA#8B;STA#80;STA#89;LDA#8C;STA#8A;STA#81;JSRVV0
1660 JSRVV29;JMPVV62
1670:VV73
1680 SEC;LDA#82;SBC#89;LDA#83;SBC#8A;BCS P+5;JMPVV69;BNEVV79
1690 LDA#82;CMP#89;BEQ P-9
1700:VV79 JSRVV61;JSR#F7D1;]
1710 $P="EINDADRES:#";P=P+L.P;[LDA#83
1720 JSR#F802;LDA#82;JSR#F802
1730 JSR#F7D1;]$P="    (J/N):";P=P+L.P;[NOP
1740:VV74 JSRVV14;CMP@#4A;BNEP+3
1750 RTS;CMP@#4E;BNEVV74;JMPVV69
1760:VV78 CMP@#7F;BNE VV80;JSRVV73
1770:VV75 LDX@0;LDY@1;LDA#89;STA#86;LDA#8A;STA#87
1780:VV76 LDA(#86),Y;STA(#86,X);INC#86;BNE VV77;INC#87
1790:VV77 LDA#86;CMP#82;BNE VV76;LDA#87;CMP#83;BNE VV76
1800 JSRVV0;JSRVV29;JSRVV14;CMP@#4A;BEQ VV75;JMPVV62
1810:VV80CMP@CH"I";BNEVV84
1820 JSRVV73;LDA#82;STA#86;LDA#83;STA#87
1830:VV82 LDY@1;LDX@0
1840:VV81 LDA#86;BNEP+4;DEC#87;DEC#86;LDA(#86,X);STA(#86),Y
1850 LDA#86;CMP#89;BNEVV81;LDA#87;CMP#8A;BNEVV81
1860 TXA;STA(#86,X);JSRVV0;JSRVV29;JSRVV14;CMP@#4A;BEQVV82
1870JMPVV62
1880:VV84 CMP@#3F;BEQP+5;JMPVV64;LDX@0;LDA S,X;STA#8000,X
1890 LDA S+256,X;STA#8100,X;INX;BNE VV84+9
1900 JSRVV14;CMP@#20;BNEP-5;LDA@12;JMPVV85
1910]PL.A8;T=P;N.;P.$6
1920 P."BEGIN:"&A'
1930 P."TABEL:"&T'
1940 P."DATA:"&S'
1950 P."EINDE:"&S+512''
1960 P."START:"&VV20'
1970 P.'"DRUK OP RETURN OM VERDER TE GAAN"';LINK#FFE3
1980 ?18=#82;RUN
```

EIGEN STATEMENTS

In de vorige ACORN NIEUWS heeft FRANS VAN HOESEL beschreven hoe je eenvoudig statements kunt assembleren ter herkenning van P-CHARME. Deze methode kan ook gebruikt worden om eigen boxen samen te stellen. Wanneer we echter naar #A000 assembleren zullen we een iets gewijzigde methode moeten toepassen, aangezien P-CHARME dan niet gebruikt kan worden. Zodoende moet de regel met PROGRAM verwijderd worden.
De eigen box zal met een interpreteerroutine moeten beginnen waarvan een listing nu volgt:

```
 10 REM INTERPRETEER ROUTINE
 20 DIMLL8;F.I=0TO8;LLI=-5;N.
 30 P.$21;P=T;GOS.a
 40 P.$6;P=T;GOS.a
 50 ?T=#40;T?1=#BF;T=LL8
 60 END
 70
 80a;!P=#0000;P=P+2
 90[
100:LL0;LDX @#FF
110:LL2;LDY #5E;DEY
120:LL3;INX;INY
130:LL4;LDA LL8,X;BMI LL6;CMP(#05),Y;BEQ LL3
140:LL5;INX;LDA LL8,X;BPL LL5;INX;LDA(#05),Y
150CMP @#2E;BNE LL2;INY;DEX;BCS LL4
160:LL6;
170CMP @#80;BEQ LL7;STA #53;LDA LL8+1,X;STA#52;STY#03;LDX#04
180JMP(#0052)
190:LL7;JMP#C558
200:LL8;];?P=#80
210 RETURN
```

We vullen de box nu als volgt:

1. T= (begin box) bijv. #A000.
2. A= (begin statements) bijv. #A100.
3. LOAD "(naam file)" (zonder sterretje)
4. RUN
5. LOAD "(naam statement file)"
6. VERWIJDER regel met PROGRAM.
7. RUN
8. indien nog een statement dan verder als bij punt 6.
   geen statement meer dan is het klaar.

Indien men de eigen box beneden #8000 wil plaatsen moet op regel 160 worden toegevoegd "AND @#7F", zodat het hoge byte van het startadres van de routine positief wordt gemaakt. Dit is noodzakelijk omdat bij het assembleren het hoge byte, met de intruktie "?T=LL0/256@#80" altijd negatief is, om door de instruktie "BMI LL6" in de interpreteer routine herkend te worden.

DOS en COS statements die met het bekende sterretje "*" beginnen volgen een eigen interpreteer routine, de command line interpreter (CLI). Het sterretje wordt in de basicinterpreter herkend, wat tot het gevolg heeft dat de DOS en COS commando's als een subroutine worden afgewerkt. Indien men een eigen box met DOS of COS commando's met het sterretje wil laten beginnen kan men de genoemde methode ook volgen. Hiervoor is echter een aangepaste interpreteerroutine noodzakelijk, welke nu volgt:

```
 10 REM  CLI ROUTINE
 20 DIMLL7;F.I=0TO7;LLI=#FFF;N.
 30 P.$21;P=T;GOSUB a
 40 P.$6;P=T;GOSUB a
 50 T=LL7
 60 END
 70a:[
 80:LL0;LDX @#FF;CLD
 90:LL1;LDY @#00;JSR#F876;DEY
100:LL2;INY;INX
110:LL3;LDA LL7,X;BMI LL5;CMP#100,Y;BEQ LL2
120:LL4;INX;LDA LL7,X;BPL LL4;INX;LDA#100,Y
130CMP@#2E;BNE LL1;INY;DEX;BCS LL3
140:LL5;
150CMP@#80;BEQ LL6;STA#9B;LDA LL7+1,X;STA#9A;CLC;LDX@#00
160JMP(#009A)
170:LL6;JMP#E3E5
180:LL7;];?P=#80
190 RETURN
```

Het vullen van de box is indentiek aan de voorgaande methode. Indien deze box niet op #A000 komt te staan, zodat P-CHARME gebruikt kan worden hoeft de regel met PROGRAM niet verwijderd te worden en kan de methode van FRANS volledig gevolgd worden. Ook hiervoor geldt dat op regel 140 "AND @#7F" moet worden toegevoegd, indien de box beneden #8000 komt testaan.

Voor het vinden van deze statements kan verschillende methode's gevolgd worden, t.w.

a. de start van deze routine in de CLI vector te zetten.
b. de CLI vector op een schakel routine te zetten.
c. door een truucje met het bootstrap printje.

a. de CLI vector.

De basic-interpreter springt met een indirekte jump naar de DOS of COS routine. Het adres van deze sprong staat op #206/#207 wat men een vector noemt. De reset routine zet dit adres op #F8EF, dat is de COS-routine, en *DOS zet dit adres op #E3E5 dit is de DOS-routine. Door nu dit sprongadres te wijzigen kan de interpreter naar de eigen gemaakte routine gaan.

Deze vector wijziging kan men in de box zelf opnemen met het volgende programma welke dan bij het opstarten gelinkt moet worden of in een eigen reset routine worden geinitialiseerd.

```
 10 PROGRAM INIT VECTORADRES.
 15 REM VOORBEELD OP #1000
 20 P.$21;P=A;GOSUB a
 30 P.$6;P=A;GOSUB a
 40 A=P
 50 END
 60a;[
 70 LDA@#00;STA #206
 80 LDA@#10;STA #207
 90 RTS;]
100 RETURN
```

Regel 70 en 80 geven dus het start adres aan de CLI vector.

b. De CLI vector op een schakel routine zetten houdt in, dat de box op een van de velden van de schakelkaart staat en dat de juiste box ingeschakeld wordt om dan daar naar toe te springen.

c. Het bootstrap truucje kan gedaan worden omdat de adressen #FFF8 t/m #FFFF gelezen worden als #7FF8 t/m #7FFF. Zie hiervoor ook ACORN NIEUWS februari 1983 blz 43-45. Op #FFF7 staat JMP (#206), welke wordt aangeroepen in de basic interpreter op #C41E. Bij het bootstrap schakelingetje staan de adressen #7FF8 en #7FF9 in RAM-geheugen en kunnen dus gewijzigd worden. Door nu hierin bijv. #7FF6 te plaatsen leest de computer dit als JMP (#7FF6) en op #7FF6/#7FF7 komt het lage en hoge byte van de eigen interpreteer routine te staan.
In een reset routine kan men deze geheugen plaatsen vullen. Bij gebruik van dit truucje kan men in de CLI routine op regel 170 JMP #E3E5 veranderen in JMP (#206).

In het kader van het onderhouds-kontrakt dat ik met mijzelf heb gesloten volgen hier nog enkele verbeteringen tov. de versie 171 (zie ook vorige ACORN NIEUWS).

```
ABEB: 07
AC93: 0D
AC9A: 0A
AC9D: 0B
ACA1: 09
ADC8: 07

A50A: 38 20 ED AF A5 57 D0 05
      20 2E C6 B0 9D 20 EC AF
      A4 59 85 98 84 99 D0 89
      20 F6 C4 20 23 AF F0 F0
      97

AA4A: 20 B6 CE 85 16 85 96 A9
      E9 20 FA A9 20 1F CC C6
      96 F0 0D 20 31 C2 A5 57
      D0 02 C6 04 B0 24 90 EC
```

Door het aanbrengen van deze verbeteringen was het helaas nodig om een extra error-nummer te introduceren:

182 Line not found in RESTORE

Het versie-nummer is inmiddels opgehoogd tot 173. Om te kijken of men deze versie bezit (veel EPROMS zullen deze verbeterde versie al bevatten) of om te kijken of deze verbeteringen correct zijn ingetypt, kan men het programma uit de vorige ACORN NIEUWS gebruiken, dat dan uiteraard moet beginnen met Z=173 ipv. 171 of 172.



oei een foutje er kan nog meer in he he opgelost

ASBK

*** ASBK-S ***

De schakelkaart is op dit moment bij zeer veel clubleden aanwezig. Dit was voor ons aanleiding om de zeer vele mogelijkheden van de schakelkaart te bekijken. Principieel zijn er een aantal zaken die terecht zouden moeten komen op de schakelkaart. Deze zijn:

1) Uitbreidings statements (dus ERROR 94 opvangen).
2) Uitbreidings functies (dus ERROR 29 opvangen).
3) Uitbreidingen welke niet ERROR 94 of 29 tot gevolg hebben als zij niet aanwezig zijn.
4) Een vervangende ERROR handler.
5) Een ander b.v. sneller Cassette Operating System (COS).
6) Veranderde read en write routines om b.v. functietoetsen mogelijk te maken.
7) De Interupt vectoren kunnen eveneens niet naar #AXXX wijzen, omdat als men schakelt tussen verschillende #AXXX pagina's dan staan deze vectoren natuurlijk verkeerd, zodat deze eveneens hier terecht komen.
8) Gewijzigde RESET, NMI en Interrupt routines.
9) De schakelsoft zelf.

Bij het ontwerp van de schakelkaart is men ervan uitgegaan dat de schakelsoft zich in het gebied #EXXX zou gaan bevinden. Voor de bezitter van de ACORN DISC DRIVE brengt dit problemen met zich mee, omdat de software voor de disc ook op #EXXX zit. In veel regio's worden momenteel de schakelkaarten omgebouwd van #EXXX naar #1XXX zodat dit ook goed aansluit bij wat er leeft in den lande. Nu we het toch over standaardisatie hebben lijkt het verstandig om eveneens afspraken te maken over het te gebruiken schakelbyte welke nodig is voor het uitlezen van de schakelbyte omdat het schakelkaart adres #BFFF write only is. Een aantal regio's hebben dit nu op #9FFF gesteld omdat iedereen hier toch al gestapelde RAM heeft en het verwant is met #BFFF. Ook bij de ASBK-I box is #9FFF gebruikt als schakelbyte.

*** STATEMENTS SCHAKELEN ***

Voor alle duidelijkheid eerst even een uitleg over Schakelkaart Operating Systemen (SOS). Deze systemen maken het mogelijk om handels of eigengemaakte eproms welke op #AXXX geadresserd zijn en welke een Command String Interpreter (CSI) hebben op #A002, inclusief testbytes #40 en #BF op #A000 en #A001, samen te laten werken zodat hierin opgenomen statements onmiddellijk herkend en uitgevoerd worden. Er zijn echter verschillende manieren om dit te bereiken. Zo kan men:

1) Succesievelijk alle ROM voeten aflopen, de een na de ander tot dat het gezochte statement is gevonden en kan worden uitgevoerd. Alle CSI worden achter elkaar geplaatst, waardoor onze ATOM erg traag wordt vooral bij statements achter in de rij. Bovendien wordt bij statements welke meer dan een keer in de CSI voorkomen steeds de eerste in de rij gepakt.
2) Een eigen CSI maken waarin je alle voorkomende statements laat voorkomen en uit statements welke meerdere malen voorkomen kies je de beste. De schakelsoft is snel (de CSI is zo kort mogelijk) maar wel omvangrijk vanwege de dubbele tabel ruimte en bovendien worden de in de aparte toolboxen aanwezige CSI niet gebruikt.

Zonde van die verloren ruimte. Ook is het zo dat bij deze methode alle eproms op een vaste plaats moeten staan. Men zou eerst een omvangrijke en langzame test kunnen laten uitvoeren om uit te zoeken waar iedere box staat maar dan vervalt ook het voordeel van de snellere CSI. Bij het ASBK-S systeem komen beide vormen voor, en het heeft dan ook zeer vele voordelen.

1) Men kan bepaalde statements met voorrang laten uitvoeren zodat deze snel zijn.
2) Statements die b.v. in direct mode en dus langzaam uitgevoerd kunnen worden, worden achter in de rij geplaatst zodat andere statements in #AXXX sneller uitgevoerd kunnen worden.
3) Het is mogelijk om het systeem "op slot" te zetten, zodat men alleen toegang heeft tot een bepaalde box om perse deze box te kunnen gebruiken.
4) Ook kan men bepalen welke van dubbele statements gebruikt zal worden, zodat steeds de beste statement kan worden gebruikt.
5) Ook als het systeem "op slot" staat zijn de statements RESET en BOX te bereiken zodat u hiermee het systeem weer aan kunt zetten.
6) U kunt zelf bepalen waar de extra uitbreidingen zich zullen bevinden b.v. op #1XXX of #7XXX.
7) De uitbreidings statements kunnen zowel BASIC als machinetaal zijn.
8) Nieuw aangeschafte eproms kunnen mits zij een CSI hebben op #A002 en test bytes #40 en #BF op #A000 en #A001 zonder meer worden bijgeprikt.

*** ANDERE VERBETERINGEN ***

Met de listing schakelsoft zijn de punten 1,2,3,4 en 9 al behandeld. Hoe u precies uitbreidings statements, functies en andere uitbreidingen kan maken zal in een volgend artikel worden verteld. De RESET routine vermeld onder 8 is in het programma functies veranderd, maar u moet hiervoor dan wel de elders in deze ACORN NIEUWS gepubliceerde hardware wijziging uitvoeren aan de boot-up. De vervangende ERROR handler geeft behalve het error nummer de gehele regel weer waarin de error optrad als de error tenminste in BASIC voorkwam, met een highlight op de plaats waar de interpretor was gebleven toen de fout optrad. Het is dus zeer eenvoudig geworden om fouten op te sporen en te verbeteren.
Een CTRL opdracht geeft nu niet meer een ERROR 94 als men hierna weer op RETURN drukt.
Er zijn echter ook een aantal dingen waarmee men moet oppassen met het ASBK-S systeem.

1) De free space pointer kan men het beste gebruiken als men een gebied wil vinden om iets in te assembleren, omdat de onder functie uitbreidingen vermelde uitbreidingsfuncties gebruik maken van ruimte boven free, zodat een hier eventueel geassembleerd programma weer wordt gewist. Dit is ook gevaarlijk in direct mode als men geen OLD heeft gegeven en wel string handling gebruikt. Om in de direct mode string handling te doen kan men dus of OLD intypen of naar een andere textspace verhuizen met b.v. SET #28.

voorbeeld assembler programma:

```
10FOR I=0 TO 1
20P=FREE
30[
40HIER UW ASSEMBLER SOURCE
50]
60NEXT I
70FREE=P
80END
```

2) Voordat u toegang heeft tot de functies en statements van het ASBK-S systeem moet u de opdracht RESET geven of de elders in deze ACORN NIEUWS gepubliceerde boot-up hardware wijzigingen maken om deze bootstrap te verkrijgen. Als u geen eigen bootstrap heeft moet u na elke druk op de BREAK toets een RESET opdracht geven.
3) De josbox moet in ROM voet 2, dit in verband met toekomstige uitbreidingsstatements welke hiervan gebruik zullen maken.

Bij RESET of na op BREAK gedrukt te hebben, met een aanwezige boot-up, verschijnt de text ACORN-ATOM al of niet met een punt erachter of u al of niet een DISC DRIVE hebt. Alle beschreven functie en statements zijn nu ter beschikking.

*** EN DAN DIT... ***

Voor functies en veranderingen welke niet ERROR 94 veroorzaken kan men niet volstaan met alleen onze 3 weg wissel in de floating point, omdat deze alleen bereikt wordt bij ERROR 94. Zo moet de interpreter gebruik maken van een vervangende error handler waarvan de pointers op #202 en #203 staan. Zoals ook al in ACORN NIEUWS nr5 van 1982 heeft gestaan kan men dit niet in direct mode veranderen zonder de OSCHRD routine te veranderen. Dit is dus de reden dat de OSCHRD vector is veranderd in het schakelsoft programma, en mag men zelf deze vector niet meer veranderen, omdat dan de interpreter niet meer werkt. Stilzwijgend is aangenomen in het voorgaande dat de 3 weg wissel in de floating point nu naar #1000 wijst.

*** FUNCTIE UITBREIDINGEN ***

* ERR
* ERL
* b
* LEFT$
* RITHGT$
* MID $
* INSTR
* [
* f
* TAB
* FREE

TS Deze functie geeft U gemakkelijk toegang tot de TEXT SPACE pointer welke zich bevindt op geheugenplaats #12 of 18
b.v. If TS =#29;PRINT" LOWER TEXTSPACE"

RESET Dit statement start het systeem. Het moet gebruikt worden na elke opstart of na een BREAK, tenzij er een bootstrap is ingebouwd in welk geval het automatisch kan.

BOX Dit statement laat U schakelen tussen de verschillende boxen.

* FREE

* REPORT
* RERUN

* Deze statements en functies zijn gelijk aan de superbasic.

*** RESET ROUTINE:

Wanneer we zelf willen bepalen waar of hoe de RESET of BREAK routine zou moeten werken zijn er een aantal manieren om dit op te lossen. Een van die manieren wordt hieronder beschreven en wordt ook toegepast in het komende ASBK-S systeem.(Schakelsoft). In deze toepassing is de break routine uit de F-rom gewijzigd. Daar deze rom niet is te wissen of te wijzigen is op de oude A-voet(IC 24) de nieuwe F-rom geplaatst.
Hiervoor moeten natuurlijk een aantal hardware wijzigingen plaatsvinden. Deze zijn echter zo eenvoudig en simpel dat nagenoeg iedereen dit zal kunnen. De Hardware wijzigingen worden in het schema duidenlijk gemaakt.

*** Nu de software:

Als we iets in de bestaande ruimte willen aanpassen geeft dit meestal problemen met de beschikbare ruimte in die routine.
In dit geval was dat dus ook het geval in de BREAK routine zodat we deze dus moesten inkorten.
Dat dit niet geheel zonder gevolgen is zal logisch zijn.
Wat is er gebeurt:

1) vanaf #21C worden #1C bytes gevuld met andere waarden dan voor de reset.Deze geheugen plaatsen zijn bij de Atom vrij of worden gebruikt voor de While Endwhile Stack,zodat dit geen gevolgen heeft.
2) De dubbele test of er #2900 geheugen zit is nu tot een enkele test teruggebracht. Dat dit geen gevolgen heeft hebben we met de drie weg wissel al geconstateerd.
3) De 8255 wordt niet direct goed geinitialiseerd maar krijgt alvorens de goede waarde te krijgen een aantal verkeerde.Het toetsenbord en de VDU blijven echter goed werken omdat de eindwaarde goed is.
4) Men kan de geheugen plaatsen #B004-#B035 niet meer gebruiken voor I/O omdat deze nu in de reset routine een waarde krijgen.
Om u gerust te stellen:
U merkt van het bovenstaande niets in de praktijk.
Het programma zal gelijk na binnen komst in de reset routine kijken of er shift-break of break was ingedrukt.
Shift Break is de oude vertrouwde ATOM BREAK maar met schift break gaat hij nu indirect springen naar adres (#1FFC).
In deze locatie kan dan het adres staan waar onze eigen break/reset routine zit.
Het leuke van het direct kijken bij binnen komst in de break routine of de shift niet is ingedrukt is dat op dat moment alle registers nog origineel zijn zodat we die eventueel ook uit kunnen lezen.De mogenlijkheden zijn dus zeer uitgebreid en door de hier gebruikte indirecte jump zeer universeel.
Wel zal er op gelet moeten worden dat in onze eigen reset/break routine,die waar dan ook staat,de poorten #B002 en #B003 goed gezet moeten worden zoals dit in de origineie Break routine gebeurt.
Dit ter voorkoming dat de hele zaak gaat hangen.

Hieronder volgen twee programma's het ene is de aangepaste break routine de andere een voorbeeld van hoe een eigen break of reset routine er uit zou kunnen zien.
Hoe gaan we te werk:
Save uw oude F-pagina.
Load het daarna op b.v #4000.
Tik het onderstaande programma in en RUN het.
Save van #4000 to #4FFF als zijnde uw nieuwe F-pagina.
Zet dit geheel in Eprom.
Maak de in het schema aangegeven hardware wijziging en uw nieuwe F-pagina is klaar voor gebruik.

```
  0REM RESET
 10DIM LL(5)
 20F.I=0TO5;LL(I)=-1;N.
 30F.I=0TO1
 40P=#4F3F;A=P
 50[
 60LDA#9FFF;AND@#5F;STA#9FFF
 70STA#BFFF;LDA#B001;BPLLL1
 80JMP(#1FFC)
 90:LL1
100LDX@#33
110:LL2
120LDA@#FF;STA#02EB,X
130LDA#FF98,X;STA#B002,X
140LDA#FF9A,X;STA#0204,X
150DEX;BPLLL2;TXS;INX
160STX#EA;STX#E1;STX#E7
170LDA@#0A;STA#FE
180JSR#F7D1;]
190?P=6;?(P+1)=#C;?(P+2)=#F;P=P+3
200$P="ACORN-ATOM";P=P+L.P
210?P=#A;?(P+1)=#A;?(P+2)=#0D;P=P+3
215[
220LDA@#82;CLI;STA#2901;CMP#2901;BNELL3
230JMP#C2B2
240:LL3
250JMP#C2B4
260]
265?P=#07;?(P+1)=#8A;P=P+2
270N.;@=0
280P."CODE VAN "&A" TOT "&P-1''
290@=8;END
```

Als volgt zou b.v. uw eigen bootup er uitkunnen zien als onderstaand programma. Er wordt gekeken of er een DOS aanwezig is ,zo ja wordt deze ingelinkt ,en de text space wordt op #2900 gezet. Tevens wordt er in plaats van de oude text ACORN-ATOM een eigen text gedisplayed.

```
 10REM ******************
 20REM **              **
 30REM ** asbk bootup **
 40REM **              **
 50REM ******************
 60DIM BB8
 70F.I=0TO8;BB(I)=-1;N.
```

```
 80F.I=1TO2
 90P=#1700
100[
110LDA@#0A;STA#FE
120LDA@#8A;STA#B003;LDA@#7;STA#B002
130BIT#B002;BVC BB0
140LDA#E000;CMP@#A9;BNEBB7;JSR#E000
150:BB7
160JSR#F7D1;];?P=12;$P+1="*** ACORN ATOM";P=P+L.P
170[LDA#E000;CMP@#A9;BNEBB4
180LDA@#2E;STA#800E
190:BB4
200JSR#FFED;JSR#FFED
210LDA@15;JSR#FFF4
220JMP#C2B2
230:BB0
240LDA@#8A;STA#B003;LDA@#07;STA#B002
250LDA@BB1/256&#FF;STA#06;LDA@BB1&#FF;STA#05;JMP#C2F2
260:BB1;]
270$P="@=1;P.$6$7' ";P=P+L.P
280$P="""BREAK """;P=P+L.P
290$P=";@=8;IF?10?2P.";P=P+L.P  300$P="""AT LINE""";P=P+L.P
310$P="!1&#FFFF"
320P=P+L.P;?P=#0D;?(P+1)=#00
330?(P+2)=#00;P=P+3
340$P="P.';E.";P=P+L.P;?P=#0D;?(P+1)=#FF;P=P+2
350P.$6;N.I
360?#1FFC=#00;?#1FFD=#17;END
```

Deze Bootup routine heeft de volgende eigenschappen:
1) poorten worden goedgezet.
2) test of DOS er is,zo ja wordt deze ingelinkt en zal er een punt achter de bootup text worden gezet.
3) REPEAT/BREAK zal een BREAK AT LINE xxx geven.
Hieronder dan de te wijzigen hardware:

*** SCHEMA ***

```
  ic 23 spin voor schakelkaart
        onderse ,ic.
  -----
  - * -                 -------  pin 20(cs) uit voet halen
  -   -                 -  *  -
  -   -                 -     -
  -   -                 -     -
  -   -                 -     ---
------------------------------
  -   -                 -     -
  -----                 -     - in oude A-voet IC24
                        -     - monteren
pin 7 uit               -     -
voet halen              -     -
                        -------
```

U ziet dat dit nogal meevalt het is maar een draadje wat gelegd moet worden tussen pin 7 van ic23 en pin 20 van ic 24 om uw nieuwe F-rom te laten werken.

Wij zelf gebruiken de #1000 pagina voor de bootup en de schakelsoft. U kunt dit natuurlijk ook op #E000(wanneer geen DOS) of #7000 doen.

*** DE ASBK-1 ***

Eindelijk na een lange tijd van debuggen en testen is hij dan klaar de ASBK-I box. Deze nieuwe toolkit gaat de Softtool II en de Program Power vervangen. Tevens zijn er een aantal nieuwe commando's toegevoegd die her en der in den lande in schakelsoft-systemen worden gebruikt. We hebben met deze box 36 commando's aan onze ATOM toegevoegd.

STARTADRESSEN ASBK - 1 BOX

COMMANDO	START	EIND	JMP ADRES
AUTO....	#A130	#A236	#A130
BEEP....	#A237	#A2A8	#A237
BELL....	#A2A9	#A2D9	#A2A9
BOX.....	#A2DA	#A30D	#A2DA
CHA.....	#AE1C	#AF1C	#AE1C
CHLOC...	#A30E	#A3D8	#A30E
CLS.....	#A3D9	#A408	#A3D9
COMPACT.	#A409	#A549	#A409
CURSOR..	#A54A	#A598	#A54A
CONT....	#ACA1	#ACE2	#ACA1
COS.....	#ADFF	#AE1B	#ADFF
DELETE..	#A599	#A675	#A59C
DUMP....	#AF84	#AFD3	#AFA1
FILL....	#A676	#A6D9	#A69C
GSCREEN.	#AF3D	#AF83	#AF3D
HEPR....	#A6DA	#A756	#A6DA
HOME....	#A757	#A762	#A757
HIGH....	#A763	#A79D	#A763
HTAB....	#ACE3	#ACF6	#ACE3
INKEY...	#A823	#A883	#A823
INVERT..	#A884	#A8D1	#A884
INFALL..	#A808	#A822	#A808
LBR.....	#A8D2	#A920	#A8D2
ON......	#A921	#AA01	#A932
PAUSE...	#AA02	#AA1C	#AA02
REPEAT..	#AA1D	#AA4D	#AA1D
SET.....	#AB17	#AB62	#AB24
SHOW....	#AB63	#AC08	#AB63
SLIST...	#AC09	#AC83	#AC09
SCREEN..	#AF1D	#AF3C	#AF1D
STOP....	#AC84	#ACA0	#AC84
SEARCH..	#AA4E	#AB16	#AA4E
INFO....	#A79E	#A807	#A79E
VTAB....	#ACF7	#AD11	#ACF7
VAR.....	#AD12	#ADDB	#AD12
ZERO....	#ADDC	#ADFE	#ADDC

NAWOORD:

Veel commando's uit de ASBK-I box komen uit handelstoolkits en zijn omgebouwd. Dit o.a. met het oog op de zero-page adressen.

MINISCHAKELKAART

Met deze minischakelkaart kan geschakeld worden tussen 2 EPROM's (2532) op #AXXX, en 4K CMOS RAM (met battery back-up), die overal in de computer geadresseerd kan worden mbv. dipswitches.
Het geheel past in de kast, en komt mbv. IC stekers ('headers') in de voetjes van IC 23, en IC 24. De twee IC's komen nu op het printje.

DE BOUW:

De print bestukken vlg. de printopstelling. Let bij het solderen vooral op dat er geen korreltjes tin tussen de spoortjes komen, en kortsluiting veroorzaken!!. Wanneer de componentenzijde klaar is moeten de IC stekers eronder gesoldeerd worden. Wanneer u headers, of flatcable connectors gebruikt, doet u er goed aan eerst door de overgebleven gaatjes draadstukjes te steken, en deze vast te solderen, als u nu de stukjes op +/-3mm onder de print afknipt, kan daar de connector op vast gesoldeerd worden. Voor de battery-back up kan zowel een eigen batterijtje,als de NICAD-cel die al op de 16K kaart,of Big benny zit gebruikt worden (in de laatste 2 gevallen R37 NIET MONTEREN (ook niet, bij het gebruik van gewone, niet oplaadbare batterijen!) en de - v/d batterij niet aansluiten (anders aardlussen)).
Nu moet alleen nog de kleine 8-pin connector aangesloten worden:

1: + van de batterij
2: - van de batterij (alleen bij gebruik van aparte batterij)
3: NRDS lijn, bijv. pin 5 v/d 8255 PIA
4: NWDS lijn, bijv. pin 36 v/d 8255 PIA
5: A15- rest: gaat naar pen 4 van IC 6
6: EPROM select, PB0 van de VIA (pin 10), of PB4 van Big Benny.
7: blijft even leeg
8: RAM in-uit PB1 van VIA (11), of PB5 van Big Benny

Als u de ram inschakelt, op een bepaald adres waar in de computer geheugen zit, moet dit geheugen ge DISABLED worden. Hiervoor moet A15 op de COMPONENTENZIJDE van de print doorgekrast worden. (zie figuur A)

Voor de mensen, die geen bootstrap in hun computer hebben, is het nu klaar!

DE BOOTSTRAP:

Een opmerking vooraf: Het kan zeker geen kwaad het artikel over de bootstrap (A.N. feb '83 pg.43-45) nog eens door te lezen!
Op het printje zitten ook een aantal poortjes, die een gedeelte v/d bootstrap voor hun rekening nemen. Om hiervan gebruik te maken moet het volgende veranderd worden:

- Het draadje wat aan het DWARSGEPLAATSTE eilandje op het bootstrap printje zit en naar pin 6 van IC 23 gaat wordt verwijderd.
- Nu wordt deze pin (6/IC23) met een klein draadje verbonden metpen 25 van de processor
- De uitgang van de 'LS 133 (d.i. pen 9) op de bootstrap print wordt nu met pen 7 van de connector van de minischakelkaart verbonden.
- Als laatste wordt nu de draad,die van pen 4 van IC 6 komt en naar het eilandje 'A15' op het bootstap printje gaat, hieraf gehaald,

en op pen 5 van de connector op de minischakelkaart gesoldeerd.
- Nu moet alleen nog link 1 (op de print gemarkeerd met een sterretje) op minischakelkaart doorgekrast worden.

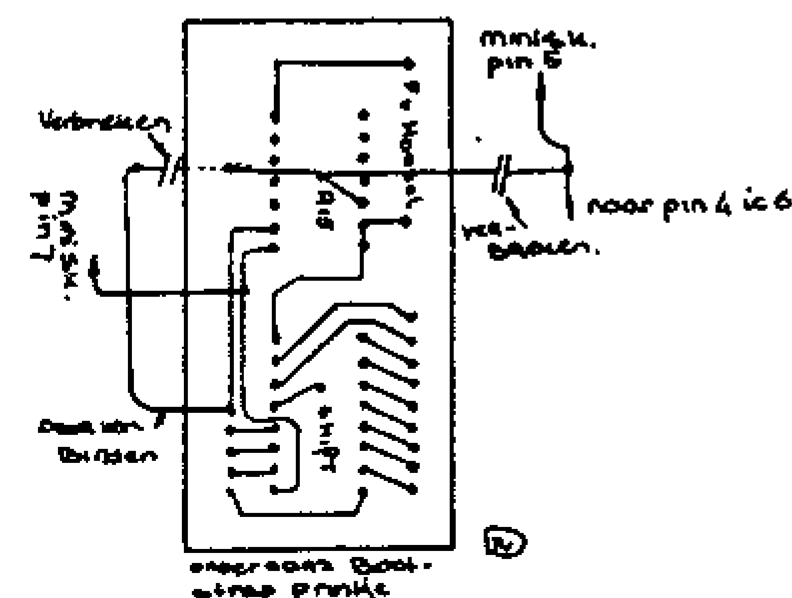
GEBRUIK:

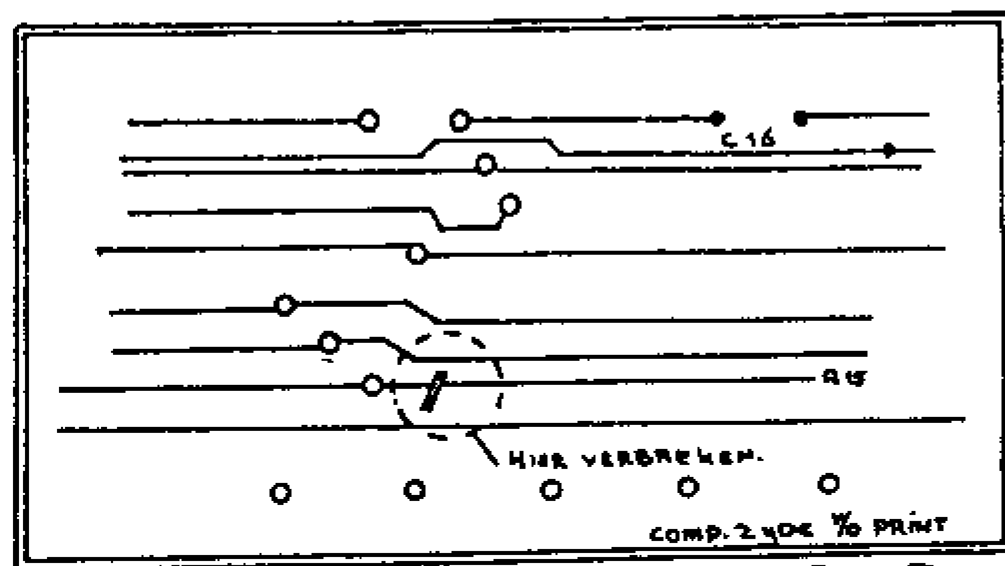
Bij het inschakelen staat de RAM uit, en is EPROM 1 geselecteerd. Met PB4 (PB0) kan omgeschakeld worden tussen EPROM1, en 2 (H: EPROM1; L:EPROM2)
Met PB5 (PB1) kan de RAM aan gezet worden (H: uit; L: aan). De plaats waar de RAM staat wordt ingesteld mbv. de dipswitches . Wanneer een dipswitch UIT staat, dan wordt dit herkend als een EEN.
RAM op #1XXX wordt zo ingesteld:A12- OFF, A13- ON, A14- ON, A15- ON.
#AXXX wordt dus zo ingesteld:A12- ON, A13- OFF, A14- ON, A15- OFF.
Als nu de RAM aangezet wordt, dan wordt de in bedrijf zijnde EPROM automatisch uitgeschakeld.
Het bovenste schakelaartje schakelt de write-protect van de ram.
NB. de beide ingangssignalen (connector pen 6 en pen 8) kunnen vanzelfsprekend ook met een schakelaartje al dan niet met massa worden verbonden om te schakelen tussen EPROM1, 2, en RAM aan, uit.

ONDERDELEN:

24 * 27K	2 * 100nF
2 * 10K	1 * LED rood 3mm
6 * 5K6	2 * BC 557
1 * 2K7	1 * DUG (aa119 o.i.d.)
2 * 1K	1 * DUS (1N4148 o.i.d.)
1 * 680 ohm	1 * BC547
1 * 470 ohm	1 * 74LS00
1 * 220 ohm	1 * 74LS 266
1 * 68 ohm	1 * 74LS156
R37: 1k (laadstroom afh.)	1 * 6-voudige DIP-switch
1 * 16pin IC steker	2 * 6116
1 * 24pin IC steker	

Er werken inmiddels 4 prototypes van het kaartje tot volle tevredenheid. Er zijn op het ogenblik een aantal printjes in een kleine serie geprduceerd. (prijs +/- F8.50) Wanneer u hiervoor belangstelling heeft, dan graag via uw regio een kaartje aan de redaktie, met het aantal benodigde printjes.



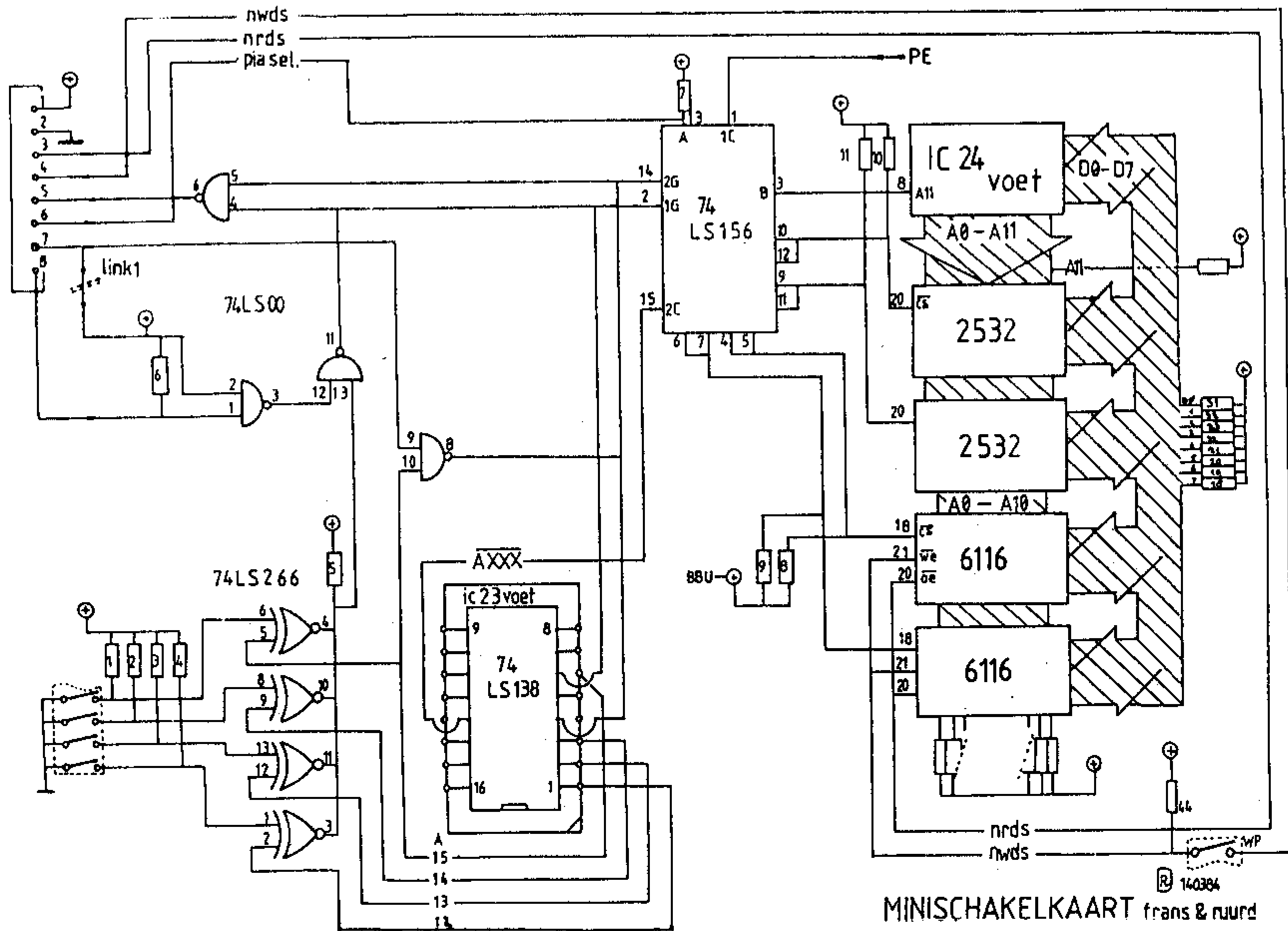


Figuur A.

VOORBEELD VAN DE BESTURING v/d MINISCHAKELKAART:

```
 10 REM SELECT A (minischakelkaart op VIA)
 20 P.$12
 30 P." RAM IN :1"'" RAM UIT:2"'
 40 P." EPROM1 :3"'" EPROM2 :4"''
 50
 60 IN." CODE" A
 70 IF A=1 GOS.a
 80 IF A=2 GOS.b
 90 IF A=3 GOS.c
100 IF A=4 GOS.d
110 E.
120
130xREM INITIALIZE
140   ?#B802=?#B802\#03;R.       :PB0+PB1: output
150
160aREM RAM AAN
170   GOS.x
180   ?#B800=?#B800&#FD;R.       :PB1: laag
190bREM RAM UIT
200   GOS.x
210   ?#B800=?#B800\#02;R.       :PB1: hoog
220cREM EPROM 2
230   GOS.x
240   ?#B800=?#B800&#FE;R.       :PB0: laag
250dREM EPROM 1
260   GOS.x
270   ?#B800=?#B800\#01;R.       :PB0: hoog

 10 REM SELECT B (minischakelkaart op PIA van Big Benny)
 20 REM regels 20-120: zie select A
130
140xREM INITIALIZE
150   ?#B403=?#B403&#FB          :select DDRB
160   ?#B402=?#B402\#30          :PB4+PB5 = output
170   ?#B403=?#B403\#04          :select DRB
180 R.
190
200aREM RAM AAN
210   GOS.x
220   ?#B402=?#B402&#DF;R.       :PB5: laag
230bREM RAM UIT
240   GOS.x
250   ?#B402=?#B402\#20;R.       :PB5: hoog
260cREM EPROM 2
270   GOS.x
280   ?#B402=?#B402&#EF;R.       :PB4: laag
290dREM EPROM 1
300   GOS.x
310   ?#B402=?#B402\#10;R.       :PB4: hoog
```


nwds
nrds
pia sel.
PE
74
LS156
IC 24
voet
D0-D7
A0-A11
A11
2532
2532
A0-A10
6116
6116
BBU
link1
74LS00
74LS266
AXXX
ic23voet
74
LS138
nrds
nwds
WP
140384
MINISCHAKELKAART frans & ruurd

8 pin connector

dipswitch

wp

adr. sel.

LS 156

LS 00

LS 266

LS 138

VOET IC 23

6116

6116

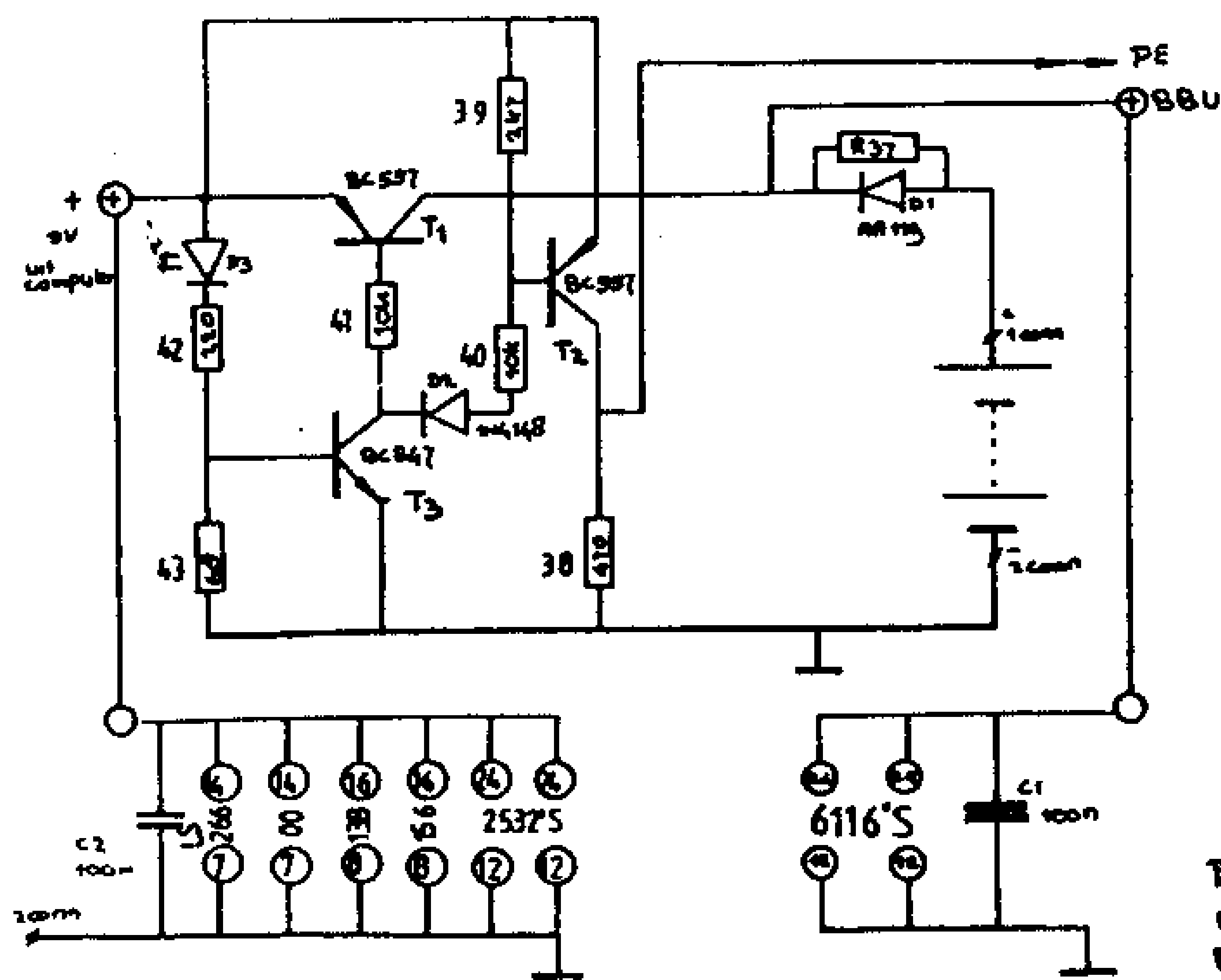
eprom 2 2532

VOET IC 24

eprom 1 2532

150484

printopstelling



INTIKKEN EN RUNNEN MAAR

In iedere uitgave zal onder bovenstaande titel listings zonder of met weinig commentaar worden gepubliceerd. Waneer listings hiervoor worden ingestuurd dan gaarne het commentaar in REM-regels.

*** ZOMBIE ***

```
  1 REM ZOMBIE
 10 H=#8000;S=#8020;E=#81C0;@=0
 20 DIM LL1,MM4,SS2;P.$21
 25 SS1=#803F;SS2=#80FF
 30 GOS.2000;GOS.2000
 42 P.$6$12;CLEAR 0
 44 P."WILT U INSTRUKTIES?"
 46 LINK LL0
 48 IF ?#90=74;GOS.4000
 50h P.$12;CLEAR 0;P."MOEILIJKHEIDSGRAAD?";LINK LL0;Q=239
 51 P.$12;CLEAR 0
 52 M=?#90-48
 54 IF M<1 OR M>9;P.$7;G.h
 56 M=M*3
 60 F.I=0 TO 31;H?I=#20;N.
 70 F.I=0 TO 31;S?I=#0F;N.
 80 F.I=#803F TO #81BF S.32
 90  ?I=#0F;I?1=#0F
100 N.
120 F.I=0 TO 31;E?I=#0F;N.
130 H?Q=#23
140 F.I=1 TO 30-M
150xR=A.R.%446
160 IF S?R<>#40;G.x
170 S?R=#0F
180 N.
190 F.I=1 TO M
200yR=A.R.%446
210 IF S?R<>#40;G.y
220 S?R=#18
230 N.;Z=M
240 GOS.3500;P."  AANTAL ZOMBIES: "Z"                    "$30;?#E1=0
250 P."       WAAR GAAT U NAAR TOE?"
260 T=Z*20;@=4
265 DO LINK LL1
270  T=T-1;P.$30T
275 U.T=0 OR (?#90>=#10 AND ?#90<=#19)
277 @=0
280 P.$30"                                    "
282 H?Q=#40
285 IF ?#90=#FF;G.350
290 G.(300+10*(?#90-#10))
300 G.f
310 Q=Q+31;G.m
320 Q=Q+32;G.m
330 Q=Q+33;G.m
340 Q=Q-1;G.m
350 G.m
```

```
360 Q=Q+1;G.m
370 Q=Q-33;G.m
380 Q=Q-32;G.m
390 Q=Q-31;G.m
400mIF H?Q=#0F;G.v
410 IF H?Q=#18;G.z
420 H?Q=#23;GOS.t
430 GOS.s;G.240
440vGOS.3500
450 P."          IN DE KUIL!!!              "
460 G.o
470zGOS.3500
480 P."          OPGEGETEN....              "
490 G.o
500fGOS.3500
510 P."          FLAUW HOOR!!!              "
520 G.o
530oP.$30"                                 "$30"NOG EEN KEER?"
540 ?#E1=0;P." ";LINK LL0
550 IF ?#90=78;G.e
560 IF ?#90=74;RUN
570 GOS.3500
580 P."       IKKE NIET BEGRIJPEN       ";G.o
600tB=H+Q
610 IF ?(B-1)=#18;?(B-1)=#40;?B=#18;G.z
620 IF ?(B+1)=#18;?(B+1)=#40;?B=#18;G.z
630 IF ?(B-32)=#18;?(B-32)=#40;?B=#18;G.z
640 IF ?(B+32)=#18;?(B+32)=#40;?B=#18;G.z
650 R.
1000sF.J=64 TO 447;V=H+J
1020 IF ?V<>#18;G.n
1030 ?V=#40
1040 X=Q%32-J%32
1050 Y=Q/32-J/32
1060 B=J;C=0;D=0
1064 IF X<>0;C=X/A.X
1068 IF Y<>0;D=Y/A.Y
1070 IF A.X>A.Y;B=B+C;G.1110
1080 IF A.X<A.Y;G.1100
1090 IF R.<0;B=B+C;G.1110
1100 B=B+D*32
1110 W=H+B
1120 IF ?W=#40;?W=#19;G.n
1130 IF ?W=#23;?W=#18;G.z
1140 IF ?W=#19;GOS.1300;G.n
1150 IF ?W=#18;?W=#19;GOS.1300;G.n
1160 IF ?W=#0F;GOS.1350;G.n
1200nIF Z=0;G.3000
1205 N.J
1240 LINK MM0
1250 R.
1300 GOS.3500
1310 P."  ZOMBIE EET ZOMBIE              ";Z=Z-1;R.
1350 GOS.3500
1360 P."  ZOMBIE VALT IN KUIL            ";Z=Z-1;R.
2000 DIM P-1
2005[:MM0
```

```
2010\ Y'S WORDEN X-EN
2020 LDX @193
2030:MM1
2040 LDA SS1,X
2050 CMP @25  (Y?)
2060 BNE MM2
2070 LDA @24  (X!)
2080 STA SS1,X
2090:MM2 DEX
2100 BNE MM1
2110 LDX @193
2120:MM3
2130 LDA SS2,X
2140 CMP @25  (Y?)
2150 BNE MM4
2160 LDA @24  (X!)
2170 STA SS2,X
2180:MM4 DEX
2190 BNE MM3
2200 RTS
2210:LL0
2220 JSR #FFE3
2230 STA #90
2240 RTS
2250:LL1
2260 JSR #FE71
2270 STY #90
2280 RTS
2290];R.
3000 GOS.3500
3010 P."OK, JIJ HEBT GEWONNEN          ";G.o
3500 P.$30''''''''''''''''';R.
4000 P.$12;CLEAR 0
4005 P.''"     *** ZOMBIE-EILAND ***"''
4010 P."U BENT ALS SCHIPBREUKELING AAN-"'
4020 P."GEKOMEN OP EEN EILAND DAT BE-"'
4030 P."WOOND WORDT DOOR MENSENETENDE"'
4040 P."ZOMBIES. DE ENIGE MANIER OM TE"'
4050 P."ONTSNAPPEN IS: DE ZOMBIES IN DE"'
4060 P."OP HET EILAND VERSPREID LIGGEN-"'
4070 P."DE VALKUILEN TE LOKKEN."
4100 P.''"(DRUK OP TOETS OM DOOR TE GAAN)";LINK #FFE3
4105 P.$12;CLEAR 0
4110 P."U KUNT DAARTOE IN ALLE"'
4120 P."RICHTINGEN BEWEGEN, ECHTER NIET"'
4130 P."MEER DAN 1 PLAATSJE PER KEER."'
4131 P."DE RICHTINGEN ZIJN DIE VAN EEN"'
4132 P."NUMERIEK TOETSENBORDJE:"'
4133 P."789   '5' IS STILSTAAN"'"456"'"123   '0' IS OPHOUDEN"'
4165 P.'"MOEILIJKHEIDSGRAAD NAAR KEUZE"'
4166 P."(EEN VAN DE CIJFERS 1 T/M 9)"'
4170 P."DE ZOMBIES KOMEN RECHT OP U AF, "
4180 P."EN U HEEFT BEPERKTE BEDENKTIJD!!"
4185 P."(MINDER ZOMBIES, MINDER TIJD)"
4190 P.''"(DRUK OP TOETS OM DOOR TE GAAN)";LINK #FFE3
4200 R.
5000eP.$12;END
```

*** CALCULATOR ***

```
 10 REM CALCULATOR
 20 REM TINY VERSCHUREN
 30 P.$12
 40 P."       *** atom calculator ***"''
 50 P."        +  OPTELLEN"'
 60 P."        -  AFTREKKEN"'
 70 P."        *  VERMENIGVULDIGEN"'
 80 P."        /  DELEN"'
 90 P."        W  WORTELTREKKEN"'
100 P."        C  CLEAR"'
110 P."        %  PERCENTAGE"'
120 P."       +%  BEDRAG + PERCENTAGE"'
130 P."       -%  BEDRAG - PERCENTAGE"'
140 P."       ↑2  TWEEDE MACHT"'
150 P."       ↑3  DERDE MACHT"'
160 P."       PI  BEREKENING MET PI"'
170 P.'"    voor starten druk return"
180 IN." "$S;GRMOD;P.$12''
190 P."   C  +%  -%   %"''
200 P."  PI  ↑2  ↑3   W"''
210 P."   7   8   9   /"''
220 P."   4   5   6   *"''
230 P."   1   2   3   -"''
240 P."   D   0   .   +"''
250 GOSUB c
260 DIMC(2)
270b%X=0
280dP.$30''''''
290 FORM=1TO19;P.$9;N.;P."             "''
300 FORM=1TO27;P.$9;N.;P."     "''
310 FORM=1TO19;P.$9;N.;P."      "'
320 FORM=1TO19;P.$9;N.;P."             "'
330 P.$30$10$9$9$9$9
340 FIF %X=0 T.P."0000000000    "'''';G.e
350 FIF%X>99999999 T.P." overflow    "'''';G.e
360 FP.%X''''
370eFORM=1TO20;P.$9;N.;P."GETAL    "'
380 FORM=1TO20;P.$9;N.;FIN.%B;IF%B=C T.G.b
390fP.';FORM=1TO20;P.$9;N.;IN."FUNCTIE "$C;IF $C="C"T.G.b
400 IF$C="+"OR$C="-"OR$C="↑2"OR$C="↑3"OR$C="W"OR$C="PI"T.G.a
410 IF$C="*"OR$C="/"OR$C="%"OR$C="+%"OR$C="-%"OR$C="↑2"T.G.g
420 IF$C="↑3"OR$C="W"OR$C="+"OR$C="-"OR$C="PI"T.G.g
430 P.$11$11;G.f
440gP.';FORM=1TO20;P.$9;N.;IN."GETAL    "'
450 FORM=1TO20;P.$9;N.;FIN.%X;IF%X=C T.G.b
460aIF $C="*"T.%X=%B*%X
470 IF $C="/"T.%X=%B/%X
480 IF $C="%"T.%X=%B/100*%X
490 IF $C="+%"T.%X=%B+(%B/100*%X)
500 IF $C="-%"T.%X=%B-(%B/100*%X)
510 IF $C="↑2"T.%X=%B*%B
520 IF $C="↑3"T.%X=%B*%B*%B
```

```
530 IF $C="W"T.%X=SQR%B
540 IF $C="+"T.%X=%X+%B
550 IF $C="-"T.%X=%X-%B
560 IF $C="PI"T.%X=%B*PI
570 G.d
580cMOVE10,15;DRAW140,15;MOVE10,40;DRAW140,40
590 MOVE10,14;DRAW140,14;MOVE109,114;DRAW109,15
600 MOVE10,65;DRAW140,65;MOVE10,90;DRAW140,90
610 MOVE10,115;DRAW140,115;MOVE10,138;DRAW140,138
620 MOVE10,114;DRAW111,114;DRAW111,15
630 MOVE10,160;DRAW140,160;MOVE10,185;DRAW140,185
640 MOVE10,161;DRAW140,161;MOVE10,186;DRAW140,186
650 MOVE10,185;DRAW10,15;MOVE140,185;DRAW140,15
660 MOVE141,186;DRAW141,14;MOVE142,186;DRAW142,14
670 MOVE9,186;DRAW9,14;MOVE8,186;DRAW8,14
680 MOVE40,160;DRAW40,15;MOVE75,160;DRAW75,15
690 MOVE110,160;DRAW110,15;R.
```

**** WOORDZOEK ****

```
 10 REM WOORDZOEK
 20 REM TINY VERSCHUREN
 30 @=1
 40 DIM R(10),Z(1),Y(3),U(3),W(10)
 50 P.$12"           *** WOORDZOEK ***"'''
 60 P."UW TEGENSPELER MAG NIET KYKEN."''
 70 P."TOETS EEN WOORD IN (MAX.10 KAR.)"'
 80 P."DRUK DAARNA DE RETURNTOETS IN."''
 90 P."TEGENSPELER MOET WOORD ZOEKEN."''
100 P."PUNTEN GEVEN AANTAL TEKENS AAN."'''
110 IN."         *** DRUK RETURN ***"'$O
120 IF $O="" T.G.130
130 P.$12
140 PLAY A1,B4,C8,D2,E1,F8,G4
150 IN."WOORD "'$R
160 IF LEN(R)>10 T.P."MAX. 10 KARACTERS";F.D=0TO200;WAIT;N.;G.130
170 M=LEN(R);X=0
180 A=?#8021;B=?#8022;C=?#8023;D=?#8024;E=?#8025
190 F=?#8026;G=?#8027;H=?#8028;I=?#8029;J=?#802A
200 P.$12
210 PLAY A2,B4,C8,D8,E4,F2,G1
220 P.$30
230 !#8092=#20202020;!#80B2=#20202020
240 !#80B6=#20202020;!#80BA=#20202020
250 !#80C0=#20202020;!#80C4=#20202020;!#80C8=#20202020
260 !#80F2=#20202020
270 IN."LETTER "'$Z
280 X=X+1;N=#2E
290 IF M>0 T.?#81EB=N
300 IF M>1 T.?#81EC=N
310 IF M>2 T.?#81ED=N
320 IF M>3 T.?#81EE=N
330 IF M>4 T.?#81EF=N
340 IF M>5 T.?#81F0=N
350 IF M>6 T.?#81F1=N
360 IF M>7 T.?#81F2=N
370 IF M>8 T.?#81F3=N
380 IF M>9 T.?#81F4=N
```

```
390 V=?#8021
400 IF V=A T.?#81CB=V
410 IF V=B T.?#81CC=V
420 IF V=C T.?#81CD=V
430 IF V=D T.?#81CE=V
440 IF V=E T.?#81CF=V
450 IF V=F T.?#81D0=V
460 IF V=G T.?#81D1=V
470 IF V=H T.?#81D2=V
480 IF V=I T.?#81D3=V
490 IF V=J T.?#81D4=V
500 P.''
510 IN."WEET U HET WOORD "$Y
520 IF $Y="NEE" T.G. 220
530 IF $Y="JA" T.G. 550
540 P.$11;G.510
550 IN."WAT IS HET WOORD "$W
560 IF $W=$R T.P.'''"U HEBT HET IN "X" BEURTEN"''';G.620
570 P."HET IS FOUT"'
580 IN."WILT U VERDER    "$U
590 IF $U="NEE"T.G.680
600 IF $U="JA" T.G.220
610 P.$11;G.580
620 !#80E0=#20202020;!#80E4=#20202020;!#80E8=#20202020
630 !#80EC=#20202020;!#80F0=#20202020
640 IN."WILT U NOG EEN KEER "$U
650 IF $U="NEE" T.G.680
660 IF $U="JA" T.G.130
670 P.$11;G.640
680 P.$12''''''''"          *** TOT ZIENS ***"'
690 END
```

**** DOBBELSTEEN ****

```
 10 REM DOBBELSTEEN
 20 REM GEBRUIK JOSBOX
 30 REM TINY VERSCHUREN
 40 A=ABSRND%6
 50 GRMOD
 60 ?#E1=0
 70 CLEAR 3
 80 P.'
 90 IF A=0 T.G.150
100 IF A=1 T.G.160
110 IF A=2 T.G.170
120 IF A=3 T.G.180
130 IF A=4 T.G.190
140 IF A=5 T.G.200
150 P.'"        O"''''';G.210
160 P."       O"'''"         O"''';G.210
170 P."       O"''"        O"''"         O"''';G.210
180 P."       O O"'''"       O O"''';G.210
190 P."       O O"''"        O"''"      O O"''';G.210
200 P."       O O"''"      O O"''"      O O"'''
210 PLOT4,36,182;PLOT5,36,83;PLOT5,84,83;PLOT5,84,182
220 PLOT5,36,182;PLOT4,40,175;PLOT5,80,175;PLOT5,80,90
230 PLOT5,40,90;PLOT5,40,175;PLOT4,36,182;PLOT5,40,175
```

```
240 PLOT4,84,83;PLOT5,80,90;PLOT4,36,83;PLOT5,40,90
250 PLOT4,84,182;PLOT5,80,175
260 P." VOLGENDE WORP"'
270 P."DRUK SPATIEBALK"'
280 P.$21
290 DIM P(-1)
300[;JSR#FFE3
310 STA#0090
320 RTS;]
330 LINK TOP
340 P.$6
350 IF $(?#0090) = #20 T.G.40
360 END
```

**** BASICTEKST PRINTER ****

```
10P.$12"***basictekst printer***"''
20IN."AANTAL TEKENS PER REGEL="T
29DIMO3
30IN.'"REGELS AANVULLEN"$O
40IN.'"AANTAL REGELS PER BLZ.="R
50IN.'"STARTADRES TEKST="S
60Q=1;E=S;P.$21$2'
70DO
80K=0;B=0;GOS.r;L=1;N=B+K
85IFB=0;B=1
90I=(T-B-K)/B;J=(T-B-K)%B
95IFI>10R?O=#4E;I=0;J=0
100DO
110IF?S=#D;S=S+3
120IF?S>#20;P.$?S;G.n
130P." ";IFI;P." "
140IFJ;P." ";J=J-1
150nS=S+1;L=L+1;U.L>N
160Q=Q+1;IFQ>R;P.$3$6'"nieuw blad aub"'$21$2;LI.#FFE3;Q=1
170S=E;P.'
180U.?(S-2)>#1F
190P.$3$6'"KLAAR"'';END
200rDO
210IF?E>#D;G.o
215E=E+3
216IF?O=#4E OR?(E-2)>#7F;U.1;R.
217IF?E=#20IFE?1=#D;E=E+1;U.1;R.
220oA=0;C=0
230sIF?E=#20;A=A+1;E=E+1;G.s
240tIFE?C>#20;C=C+1;G.t
250IFA+B+C+K>T;U.1;R.
260B=B+A;K=K+C;E=E+C
270U.0
300DIT PROGRAMMA PRINT TEKSTEN
310ZOALS DIT ZONDER DE REGELNUMMERS UIT EN NEGEERT DE RETURNS.
320DE REGELLENGTE KAN ZELF BEPAALT WORDEN EVENALS OPVULLEN TOT
330DE RECHTERKANTLIJN.
350
360EEN REGEL ALS 350 (1 SPATIE NA NUMMER) FORCEERT EEN NIEUWE
370REGEL.
```

*** SPLITSCREEN ***

```
  0 REM SPLITSCREEN 2-1-'84
 10 REM ROB VAN DORT
 20 REM DIT PROGRAMMA BEVAT DE ASSEMBLER-ROUTINES OM HET
 30 REM STANDAARD ATOM-SCHERM IN EEN BOVEN- EN ONDERSCHERM
 40 REM TE VERDELEN
 50 REM BIJVOORBEELD TE GEBRUIKEN BIJ KOMMUNIKATIE TUSSEN
 60 REM TWEE ATOM'S
 70 REM DE SCHERMEN SCROLLEN AUTOMATISCH EN ONAFHANKELIJK
 80 REM VAN ELKAAR
 90 B=#8000;REM STARTADRES BOVENSCHERM
100 O=#8100;REM STARTADRES ONDERSCHERM
110 DIM CC(19),A(64)
120 FOR I=0 TO 19;CC(I)=#FFFF;NEXT I
130 CC(1)=#94;REM BOVENCURSOR
140 CC(2)=#95;REM ONDERCURSOR
150 T=#90;REM  SAVE ACCU
160 S=#91;REM SCHERMPOINTER: 0=BOVEN; 2=ONDER
170 W=#93;REM KORTE DUUR SAVE X-REGISTER
180 Y=#96;REM SAVE Y-REGISTER
190 X=#97;REM SAVE X-REGISTER
200
210 DIM LL(40),HH(40)
220 FOR I=0 TO 40;LL(I)=#FFFF;HH(I)=#FFFF;NEXT I
230
240 PRINT $12$21;REM SCHERM UIT
250 FOR I=1 TO 2
260 P=#7000;REM STARTADRES MACHINECODE
270[
280:LL0 \INITIALISATIE
290 LDA@32;LDX@0
300:LL1 STA B,X;STA O,X;DEX;BNE LL1 \WIS DE SCHERMEN
310 STX CC1;STX CC2 \CURSORS LINKSBOVEN
320 LDA B,X;EOR@#80;STA B,X
330 LDA O,X;EOR@#80;STA O,X
340 RTS
350
360:CC0 \INVERTEER CHAR ONDER CURSOR:CURSOR NAAR X-REGISTER
370       PHA;STX W
380       LDA S;BNE CC3
390       LDX CC1;LDA B,X;EOR@#80;STA B,X;PLA;LDX W;RTS
400      :CC3 LDX CC2;LDA O,X;EOR@#80;STA O,X;PLA;LDX W;RTS
410
420:CC4 \CURSOR VAN JUISTE SCHERM NAAR X-REGISTER
430       PHA
440       LDA S;BNE CC5
450       LDX CC1;PLA;RTS
460      :CC5 LDX CC2;PLA;RTS
470
480:CC6 \X-REGISTER NAAR JUISTE CURSOR
490       LDY S;BNE CC7
500       STX CC1;RTS
510      :CC7 STX CC2;RTS
520
```

```
530:CC8 \ACCU NAAR JUISTE SCHERM,X
540      LDY S;BNE CC9
550      STA B,X;RTS
560     :CC9 STA D,X;RTS
570
580:CC10 \SCROLL JUISTE SCHERM
590      PHA;STX W
600      LDA S;BNE CC11
610      \SCROLL BOVENSCHERM
620      LDX@32
630     :LL3 LDA B,X;STA B-32,X;INX;BNE LL3
640      LDA@32;LDX@224
650     :LL4 STA B,X;INX;BNE LL4 \ONDERSTE REGEL MET SPATIES
660      PLA;LDX W;RTS
670     :CC11 \SCROLL ONDERSCHERM
680      LDX@32
690     :LL6 LDA D,X;STA D-32,X;INX;BNE LL6
700      LDA@32;LDX@224
710     :HH8 STA D,X;INX;BNE HH8 \ONDERSTE REGEL MET SPATIES
720      PLA;LDX W;RTS
730
740 \ENTREE PRINTCHAR ROUTINE
750
760:LL7 STX X;LDX@0;STX S;JMP LL27 \ASCII NAAR BOVENSCHERM
770:LL8 STX X;LDX@2;STX S             \ASCII NAAR ONDERSCHERM
780
790:LL27 STA T \BEWAAR ACCU
800       STY #96 \BEWAAR Y-REGISTER
810       CMP@2;BNE LL9 \CTRL-B \PRINTER AAN
820       NOP;NOP;NOP;JMP LL26 \zie opmerking
830:LL9   CMP@3;BNE LL10 \CTRL-C \PRINTER UIT
840       NOP;NOP;NOP;JMP LL26 \zie opmerking
850:LL10 CMP@6;BNE LL11 \CTRL-F \SCHERM AAN
860       JMP LL26
870:LL11 CMP@7;BNE LL12 \CTRL-G \BELL
880       JSR#FD1A;JMP LL26
890:LL12 CMP@8;BNE LL13 \CTRL-H \BACKSPACE
900       JSR CC4;CPX@0;BEQ HH6
910       JSR CC0;DEX;JSR CC6;JMP LL25
920      :HH6 JSR#FD1A;JMP LL26
930:LL13 CMP@9;BNE LL14 \CTRL-I \HOR. TAB
940       JSR CC4;CPX@255;BEQ HH4
950       JSR CC0;INX;JSR CC6;JMP LL25
960      :HH4 JSR CC10 \SCROLL
970       JSR CC4;TXA;AND@224;TAX;JSR CC6;JMP LL25
980:LL14 CMP@10;BNE LL15 \CTRL-J \LINEFEED
990       JSR CC0
1000      JSR CC4;TXA;CMP@223;BCS HH1
1010      ADC@32;TAX;JSR CC6;JMP LL25
1020     :HH1 JSR CC10;JMP LL25 \SCROLL
1030:LL15 CMP@11;BNE LL16 \CTRL-K \VERT. TAB
1040      JSR CC4;CPX@32;BCC HH7
1050      JSR CC0;TXA;SEC;SBC@32;TAX;JSR CC6;JMP LL25
1060     :HH7 JSR#FD1A;JMP LL26
1070:LL16 CMP@12;BNE LL17 \CTRL-L \WIS SCHERM
1080      LDA@32;LDX@0
1090     :HH3 JSR CC8;DEX;BNE HH3
```

```
1100        JSR CC6;JMP LL25
1110:LL17 CMP@13;BNE LL18 \CTRL-M = RETURN
1120        JSR CC0
1130        JSR CC4;TXA;AND@224;TAX;JSR CC6;JMP LL25
1140:LL18 CMP@21;BNE LL19 \CTRL-U \SCHERM UIT
1150        NOP;NOP;NOP;JMP LL26 \zie opmerking
1160:LL19 CMP@30;BNE LL20 \CTRL-↑ \CURSOR HOME
1170        JSR CC0;LDX@0;JSR CC6;JMP LL25
1180
1190:LL20 CMP@32;BCS LL21
1200        JMP LL26
1210:LL21 CMP@127;BNE LL23 \DELETE
1220        JSR CC4;CPX@0;BEQ HH5
1230        JSR CC0;DEX;LDA@32;JSR CC8;JSR CC6;JMP LL25
1240       :HH5 JSR#FD1A;JMP LL26
1250
1260    :LL23 \HET TE PRINTEN CHAR IS EEN LEESBAAR ASCII TEKEN
1270
1280        CLC;ADC@32;BMI LL2 \ZET ASCII OM
1290        EOR@96              \NAAR BEELDSCHERMCODE
1300       :LL2
1310        JSR CC4;JSR CC8;INX;BNE LL24 \PRINT :CURS.NAAR RECHTS
1320        JSR CC10;JSR CC4;TXA;AND@224;TAX \SCROLL:CU. START
1330       :LL24 JSR CC6
1340:LL25 \AFSLUITING
1350        JSR CC4
1360        JSR CC0 \INVERTEER CHAR. ONDER CURSOR
1370:LL26
1380        LDX X;LDY Y;LDA T;RTS  \ VERLAAT PRINT-ROUTINE
1390]
1400 NEXT I
1410 @=4;PRINT #6'"CODE VAN #"&LL0" TOT #"&P'
1420
1430 INPUT '"DEMONSTRATIE GEWENST (J/N) "$A;IF ?A()CH"J" END
1440
1450 DIM DD(6);FOR I=0 TO 6;DD(I)=#FFFF;NEXT I
1460 DIM Q(30)
1470 PRINT $21;FOR I=1 TO 2;P=Q
1480[
1490:DD0 \ENTRY DEMO
1500 CMP@CH" ";BNE DD3 \GEEN SPATIE
1510 LDA#E7;BNE DD1 \LOCK IN
1520 LDA#B001;CMP@127;BNE DD2 \SHIFT NIET IN
1530:DD1 \INVERTED SPACE
1540 LDA@128;JMP DD3
1550:DD2 \SPACE
1560 LDA@32
1570:DD3 \PRINTCHAR
1580 CMP@CH"'"
1590 BCS DD4 \ALS LOWERCASE
1600 JSR LL7 \PRINT OP BOVENSCHERM
1610 RTS \KLAAR
1620:DD4 \LOWERCASE CHAR
1630 JSR LL8 \PRINT OP ONDERSCHERM
1640 RTS \KLAAR
1650:DD5 RTS \KLAAR MET DEMO
1660:DD6 LDA@13;JSR LL8;LDA@10;JMP LL8 \(CR) OP ONDERSCHERM
1670]
```

```
1680 NEXT I
1690
1700 LINK LL0;REM INITIALISEER SCHERMEN
1710 ?#208=DD0×256;?#209=DD0/256;REM DD0 IS NIEUWE WRCHVECTOR
1720 PRINT "HhAaLiLiDo"'
1730 LINK DD6
1740 PRINT "IK, UW TROUWE ApTrOiMm GEHOORZAAM NOG PRIaaMtAom"
1750 LINK DD6
1760 PRINT "shiftspatiegeeft"
1770 PRINT "eengeinverteerdespatie";LINK DD6
1780 PRINT "terugnaarstandaardmetbreak"
1790 LINK DD6
1800 PRINT '"ALLEEN DE CURSOR-CONTROL EN COPY TOETSEN"
1810 PRINT " KUNT U NIET MEER GEBRUIKEN"''
1820 LINK DD0
1830 END
1840
1850 opmerking: INDIEN BEDIENING GEWENST, VERVANG DAN
1860            DE NOP-INSTRUCTIES DOOR JSR#FE52
1870
1880 gebruiksaanwijzing:
1890
1900 PRINTEN NAAR BOVENSCHERM: ASCII IN ACCU; JSR LL7
1910
1920 PRINTEN NAAR ONDERSCHERM: ASCII IN ACCU; JSR LL8
```

**** MATRIX VERMENIGVULDIGING****

```
  10REM MATRIX
  20REM (VERMENIGVULDIGING)
  30REM E.KUCKARTZ
  40REM WIMMERSTRAAT 37
  50REM 6471 AA EYGELSHOVEN
  60REM 045-352643
 100DIMB1,C11;P.$12;@=0;?16=C;?17=C&#FFFF/256
 110P."SENTINEL ='0'"'
 115P.'';$C="P.$11;G.120"
 120IN."AANTAL RIJEN     MATRIX 1"O;IFO<=0;E.
 125$C="P.$11;G.130"
 130IN."AANTAL KOLOMMEN MATRIX 1"P;$C="P.$11;G.140"
 140IN."AANTAL RIJEN     MATRIX 2"Q;$C="P.$11;G.150"
 150IN."AANTAL KOLOMMEN MATRIX 2"R
 160IFP<>Q;P.'"PRODUKT BESTAAT NIET"';G.115
 165DIMV9
 170DIMMM(O*P),NN(Q*R),PP(O*R)
 180P.''"EERSTE MATRIX :"'
 190I=1;J=0;$C="P.$11;G.192"
 191J=J+1
 192P."("I","J")=";IN.A;MM(I*P-P+J)=A;IFJ<P;G.191
 193IFI<O;I=I+1;J=0;G.191
 200P.''"TWEEDE MATRIX :"'
 210I=1;J=0;$C="P.$11;G.212"
 211J=J+1
 212P."("I","J")=";IN.A;NN(I*R-R+J)=A;IFJ<R;G.211
 213IFI<Q;I=I+1;J=0;G.211
 220DOP."MATRIX 1 :"'';@=5
 230F.I=1TOO;P.';F.J=1TOP;P.MM(I*P-P+J);N.;N.;P.'
```

```
240P.''"MATRIX 2 :"''
250F.I=1TOQ;P.';F.J=1TOR;P.NN(I*R-R+J);N.;N.;@=0
260IN.''"WILT U IETS VERANDEREN  (J/N)"$B;IF$B="N";G.a
265$C="P.'';G.270"
270IN.'"WELKE MATRIX"M;IFM<>1A.M<>2;P."FOUTIEVE INVOER"';G.a
280IN."WELKE RIJ"S,"WELKE KOLOM"K,"NIEUWE WAARDE"N
290IFM=1 A.S<=O A.K<=P;MM(S*P-P+K)=N;G.a
300IFM=2 A.S<=Q A.K<=R;NN(S*R-R+K)=N;H.a
310P.'"FOUTIEVE INVOER"';G.a
320aU.$B="N"
330F.I=1TOO;F.J=1TOR;PP(I*R-R+J)=0;N.;N.
340F.I=1TOO;F.J=1TOR;F.L=1TOP
350PP(I*R-R+J)=PP(I*R-R+J)+MM(I*P-P+L)*NN(L*R-R+J);N.;N.;N.
360P.''"MATRIX 1 * MATRIX 2 :"'';@=5
370F.I=1TOO;P.';F.J=1TOR;P.PP(I*R-R+J);N.;N.;@=0
380P.'';LI.#FFE3;P.$12;@=0;G.115
```

**** STATEMENT CODE ****

```
 10 PROGRAM STATEMENT-CODE
 20 REM FRANS VAN HOESEL
 30 REM EN KLAAS DE RAAD
 40 REM INVOER CODE (BEGINADRES),(EINDADRES)
 50 REM INSTRUCTION ?:
 60 REM BIJV  JMP #C558
 70J=316;DIMLLJ;F.I=0TOJ;LLI=#FFF;N.;DIM C10
 80
 90P.$21;P=A;GOSUB a
100P.$6;P=A;GOSUB a
110$T="CODE";T=T+LEN(T)
120 ?T=LL0/256@#80;T?1=LL0%256;T?2=#80;T=T+2
130A=P;T!1=A
140END
150
160a[
170:LL3;BRK
180:LL0;LDA#06;ORA@#01;BEQLL3;LDX@#03
190STX#04;JSR#C78B;JSR#C231;JSR#C4E1
200:LL264JSR#FFED;JSR#F7D1
210];$P="INSTRUCTION";P=P+L.P;[
220NOP;JSR#CD09;LDA@#02;STA#59;CPY@#44;BPLLL130;DEC#59
230:LL130LDY@#40;JSR#F876;LDA#100,Y;CMP@#55;BNELL131
240SEC;LDA#101,Y;SBC@#30;STA#BFFF;JMPLL264
250:LL131CMP@#23;BNELL134;LDA@#01;STA#06;LDA@#40
260STA#05;LDA@#00;STA#03;JSR#C3C8;LDX@#03
270:LL132LDA#52,X;STA#140,X;DEX;BPLLL132;JMPLL265
280:LL133JMP#C55C
290:LL134LDA@#5B;STA#100;LDX@#FF
300:LL135INX;LDA#140,X;STA#101,X;CPX@#37;BPLLL133
310CMP@#0D;BNELL135;LDY@#00
320:LL137LDALL263,Y;STA#101,X;INX;INY;CPY@#0B
330BNELL137;LDA@#40;STA#331;LDA@#01;STA#34C
340LDA@#15;JSR#FFE9;JMP#C2D5
350:LL316LDA@#06;JSR#FFE9
360LDA#331;SEC;SBC@#40;STA#59
370:LL265LDX@#04
380JSR#C3CB;INX;LDY@#54;JSR#C3CE;LDX@#00;STX#04;LDX@#52
```

```
390:LL138LDY@#00
400:LL139LDA#140,Y;CMP(#52),Y;BNELL141;INY;CPY#59
410BNELL139;LDA#53;JSR#F802;LDA#52;JSR#F802;JSR#FFED;JSR#C504
420:LL141JSR#FA08;BNELL138;JMPLL264
430:LL263;]
440B=LL316;STRB,C;$P=";]LI.";P=P+L.P;$P=$C;P=P+LENP-6;?P=13
450 REM ASSEMBLEREN BOVEN #2710 LENP-6 WORDT LENP-5
460P=P+1
470 RETURN
```

**** SOFTPRINT BUFFER ****

```
 10 REM SOFTPRINT-BUF
 20 GOTO 270
 60***VERSIE 1.0***
 80S O F T P R I N T - B U F
100 DIT PROGRAMMA SIMULEERT EEN BUFFER VOOR DE PARALLEL-INTER-
110 FACE. MET DEZE ROUTINE IS HET MOGELIJK OPTIMAAL GEBRUIK TE
120 MAKEN VAN DE PROCESSOR-TIJD. INDIEN DE AANGESLOTEN RANDAP-
130 PARATUUR EEN "BUSY" SIGNAAL GEEFT, WORDT DE TE VERSTUREN
140 DATA OPGESLAGEN IN EEN BUFFER. DEZE BUFFER WORDT GEDUMPT ZO
150 GAUW "NOT-BUSY" GELDT. RAAKT E BUFFER VOL, DAN ZAL DE PRO-
160 CESSOR GEBLOKKEERD WORDEN, TOTDAT DE LEIDING WEER VRIJ GE-
170WORDEN IS. DE BUFFER-GROOTTE IS AFHANKELIJK VAN DE BUFFERS
180 IN DE RAND-APPARATUUR.
190
200 (C) COPYRIGHTS 1983 RESERVED BY     hasproductions
270 N=80;REM BUFFER-GROOTTE
280 Q=#2800;REM PLAATS VAN ROUTINE
290 R=#9F00;REM KARAKTER-BUFFER
300 B=#9FFF;REM BUFFER-NIVEAU-INDIKATOR
310 T=#9FFE;REM TIJDELIJK X-REGISTER
320
340 P.$21
350 DIM LL(4)
360 FOR I=1 TO 2
370 P=Q
380 [
390:LL0 PHA;STX T;JSR #FE55;LDX B;STA R,X;INC B;CPX @N-1;BEQLL2
400:LL1 BIT #B801;BMI LL4
410:LL2 LDA R(0);JSR #FEFB;LDX @0
420:LL3 LDA R+1,X;STA R,X;INX;CPX B;BNE LL3;DEC B;BNE LL1
430:LL4LDX T;PLA;RTS
440]
450 NEXT
460 P.$6
470 ?#208=LL0;?#209=LL0&#FFFF/256;REM DEFINIEER NIEUWE ROUTINE
480
500 REM *****INITIALISATIE****
510 ?B=0
520 END
540
550 REM   TESTROUTINE
560 PRINT $12$2
570 S=#3C00
580 $S="HALLO, DIT IS EEN TEST: KIJKEN OF HET WERKT!.."
590
```

```
600 FOR I=1 TO 5
610  FOR J=1 TO 5
620   FOR K=0 TO LEN(S)-1
630   REM TEST BUFFER OVERFLOW
640   REM FOR W=1 TO 2;WAIT;NEXT
650   PRINT $(K?S)
660   NEXT K
670  NEXT J
680  PRINT '
690 NEXT I
700 P.$3
710 END
```

**** DISK COPIEEREN ****

```
 10 PROGRAM COPY
 20
 30 DIM LL8;FOR I=0 TO 8;LL(I)=999;NEXT I
 40 PRINT $21;GOSUBa
 50 PRINT $6 ;GOSUBa
 60 END
 70aP=#8200
 80[LDA @1;STA #D1;JSR LL4;CMP @#1B;BNE LL0;RTS;
 90:LL0;LDA @#35;JSR #E7D2
100LDA @#0D;JSR #E809;LDA @#14;JSR #E809;LDA @#05;JSR #E809
110LDA @#2A;JSR #E809;LDA @0;STA #A8;STA #A9;
120:LL1;LDY @#43;LDA @0;STA #9C;STA #A0;LDA @#24;STA #9D
130LDA #A8;STA #A3;CMP @#90;BNE LL2;LDY @0;JMP #E713;
140:LL2;CMP @#4F;BNE LL3;LDY @#41;
150:LL3;STY #A1;LDA #A9;STA #A2
160JSR LL4;JSR #E4A3;JSR #E226;LDA #A9;STA #A2;LDA #A8
170STA #A3;LDA @#24;STA #9D;JSR LL7;JSR #E6B0;JSR #E226
180CLC;LDA #A1;ADC #A8;STA #A8;BCC P+4;INC #A9;JMP LL1;
190:LL4;LDA #D1;BEQ LL6;JSR LL8;];$P="ORIGINAL DISK";P=P+13;[
200DEC #D1;
210:LL5;JSR #FFED;JSR #FFE3;PHA;JSR #E75B;PLA;RTS
220:LL6;JSR #E75B;RTS;
230:LL7;LDA #D1;BNE LL6;JSR LL8;];$P="COPY";P=P+4;[
240INC #D1;BNE LL5;
250:LL8;JSR #E016;];$P="INSERT ";P=P+7;[NOP;JMP #E016
260]
270 RETURN
280
290  DIT PROGRAMMA KAN SCHIJVEN COPIEREN EN IS BEDOELD VOOR
300  DOS-GEBRUIKERS MET 1 DRIVE.
310  ER IS GEHEUGEN NODIG TOT #6700. HIERDOOR HOEFT MEN
320  SLECHTS ZES KEER VAN SCHIJF TE WISSELEN OM DE COMPLETE
330  SCHIJF TE COPIEREN.
340
350  HET PROGRAMMA MOET GE-ASSEMBLEERD WORDEN NAAR #8200
360  EN HET MACHINETAAL GEDEELTE KAN OP DE UTILITY-SCHIJF
370  WORDEN GE-SAVED. MET *COPY KAN DEZE ROUTINE DAN WORDEN
380  OPGESTART.
390
400                                                      FRANS
```

**** PRIEMGETALLEN ****

Hierbij een programma, in P-CHARME BASIC geschreven, wat priemgealten uitzoekt uit een array. De methode waarmee dit gebeurt staat in de wiskunde bekend als 'de zeef van Eratosthenes', de methode gaat als volgt:

1. zet alle getallen van 1 tot n (de bovengrens) in een array
2. schrap nu alle 2-vouden, behalve 2 zelf, daarna de 3-vouden, behalve 3, nu de 5-vouden (4, een 2-voud was al geschrapt) enz.
3. Ga bij een max. van 1000 door tot alle 31-vouden.
4. lees de overgebleven getallen uit.

Het programma is GESTRUCTUREERD, en maakt (dus) goed gebruik van PROCEDURES.

```
 10 PROGRAM PRIEMGETAL
 15 REM R. van Drunen
 20
 30 DIM RR(1000)
 50 PROC VUL(N),I
 60  FOR I=1 TO N
 70   RR(I)=I
 80  NEXT
 90 PEND
100
110 PROC PRIEM(N),I,T
120  RR(1)=0
130  T=2
140  DO
150    FOR I=T+1 TO N
160     IF RR(I)%T=0 THEN RR(I)=0
170    NEXT
180    DO
190      T=T+1
200    UNTIL RR(T)()0
210  UNTIL T)31
220 PEND
230
240 PROC SCHRIJF(N),I,T
250  T=0
260  FOR I=1 TO N
270   IF RR(I)()0 THEN P.RR(I);T=T+1
280  NEXT
290  P.''"AANTAL GETALLEN:"T
300 PEND
310
320 PROC INVOER (:A)
330  P.$12
340  P."======PRIEMGETALLEN======"'
350  P."GEEF DE BOVEGRENS"
360  DO
370    IN.'"TUSSEN 0 EN 1000:"A
380  UNTIL A)0 AND A(1000
390 PEND
400
410 REM HOOFD PROGR.
430 INVOER(X)
440 VUL(X)
450 PRIEM(X)
460 SCHRIJF(X)
480 END
```